# TRANSITION TO MBSE AND OPM INTRO

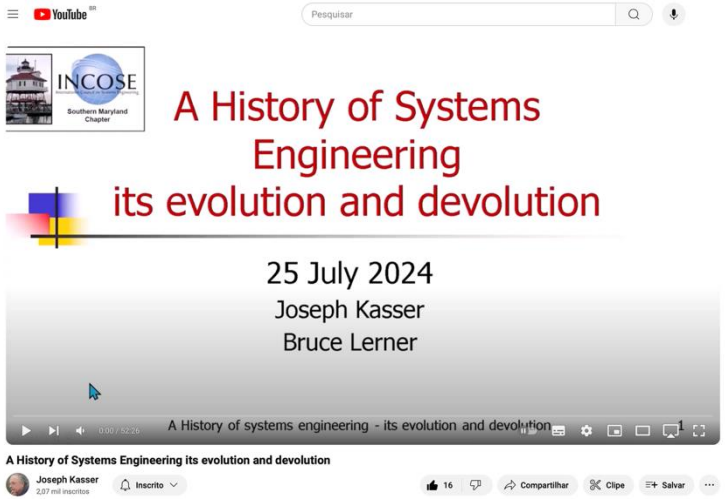
| WEEK | CLASS ACTIVITY | REF | INDIVIDUAL | W | GROUP | W |
|---|---|---|---|---|---|---|
| 1<br>28/Jul | Course Structure and Initial Definitions<br>Systems Engineering Review | [illegible] | IA-01 - Reading and Conceptual Questions (10) | 10% | | 0% |
| 2<br>04/Aug | Classical Systems Engineering Diagrams<br>(IDEF-0/N2/eFFBD/DFD) | [4]<br>*papers | [illegible] | 0% | GA-02 - Prepare a representation of your system using classical Diagrams | 50% |
| 3<br>11/Aug | Transition from Legacy to MBSE<br>MBSE Methodologies<br>MBSE Languages | [5] [7]<br>*papers | IA-03 - Reading and Conceptual Questions (10) | 10% | | 0% |
| 4<br>18/Aug | OPM - Basic | [6] | IA-04 - Exercises | 10% | | 0% |
| 5<br>25/Aug | OPM - Extended | [6] | IA-05 - Exercises | 10% | | 0% |
| 6<br>01/Sep | OPM - Group Presentation | | IA-06 | 0% | G6 - Prepare a presentation of your system using OPM | 50% |
| 7<br>08/Sep | SysML Introduction<br>(bdd/ibd) | [7] | IA-07 - Exercises | 10% | | 0% |
| 8<br>15/Sep | P1 - Conceptual Questions and Case | [1] [2] [3] [4] [6] | IA-08 - Questions and a mini-case | 50% | GA-08 - | |
| | | | | 100% | | 100% |

# Motivation

# Teasing: https://www.youtube.com/watch?v=Gmnc-78TUuM

Nine perspectives:

1. The introductory phase: the **early systems and the start of systems engineering** postgraduate education
2. Changes in the **definitions** of systems engineering: the changes in a sample of definitions of systems engineering between the 1950's and 2024.
3. Changes in the **application** of the systems approach in systems engineering: starting with the General Systems Theory, the changes in the meaning of the systems approach.
4. Changes in the systems engineering **tools**: a look at the tools of systems engineering in the 1950's and 1960's and different set of tools in the early 2002's before Model-based Systems Engineering (MBSE).
5. Changes in the systems engineering **roles**: a look at how the roles of the systems engineer changed between 1969 and 2024 with samples from1969, 1988, 1994, 1996, 1997, 2000, 2017, 2019 and 2024.
6. The two systems engineering **paradigms**: a brief overview of the original "A" paradigm and the devolved currently widely-practiced "B" paradigm which seems to have burst on the scene in the 1990's.
7. The early "**Standards**" for systems engineering: a brief look at MIL-STD 499, EIA-632, IEEE-1220 and ISO-IEC 15288:2002 showing why they are not actually standards for the performance of systems engineering.
8. Historical sketch of INCOSE: highlights a few milestones from its beginning as the **National Council on Systems Engineering** to the introduction and singular focus on MBSE.
9. The nine perspectives of systems engineering: shows how the differences in the contents of textbooks, and journal and conference papers were grouped into nine perspectives.



SPICY SE

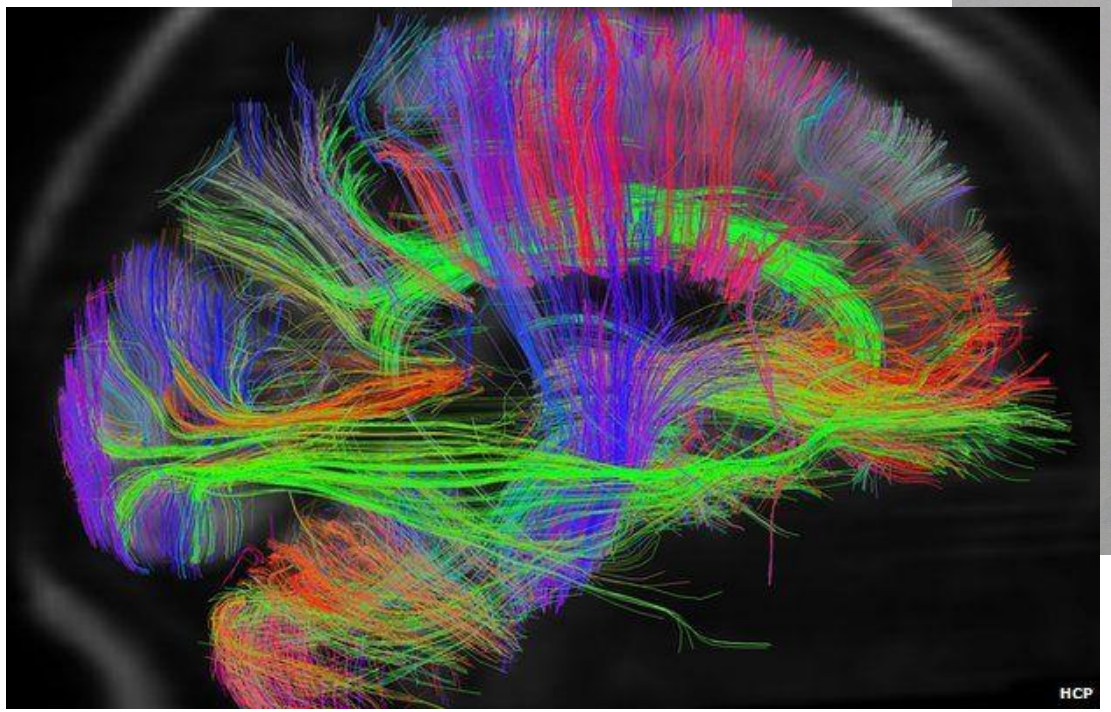
"Systems Engineering tools look like they are from the 90s!"

https://spicy-se.com

# Language Perspective

# Reducing to a function - we are standard recognizers

https://www.psychologytoday.com/blog/the-athletes-way/201311/what-is-the-human-connectome-project-why-should-you-care

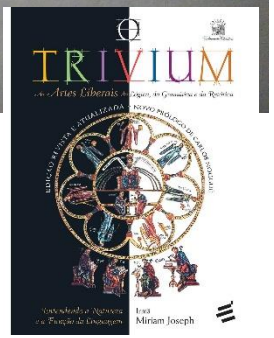



feita de tijolos, pedras ou de

## FUNÇÃO DA GRAMÁTICA

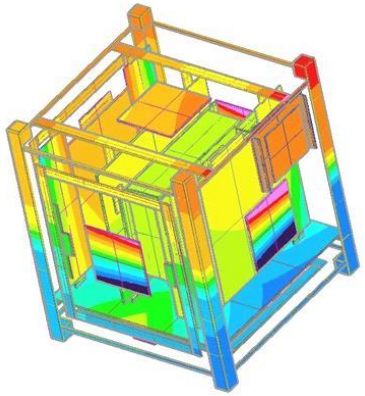
A função fundamental da gramática é estabelecer leis para relacionar símbolos de modo a expressar pensamento. Uma frase expressa um pensamento – uma relação de ideias – numa declaração, numa pergunta, numa ordem, num desejo, numa prece ou numa exclamação. Símbolos categoremáticos são aqueles que são relacionados; símbolos sincategoremáticos são os meios de relacioná-los; a oração é a relação mesma.

As regras para relacionar símbolos regem três operações gramaticais: substituir símbolos equivalentes, combinar símbolos e separar símbolos.

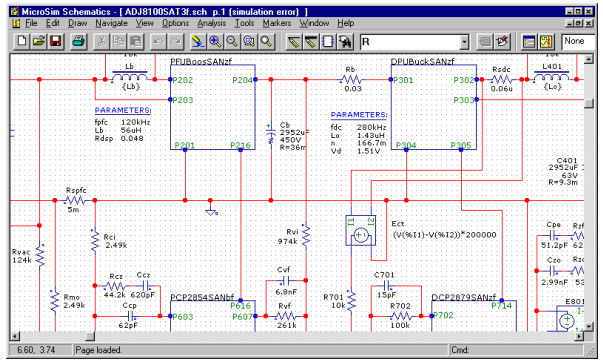


# Each engineering has its own language

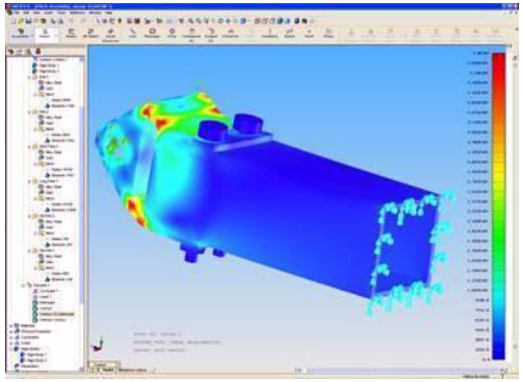
Thermal Eng.



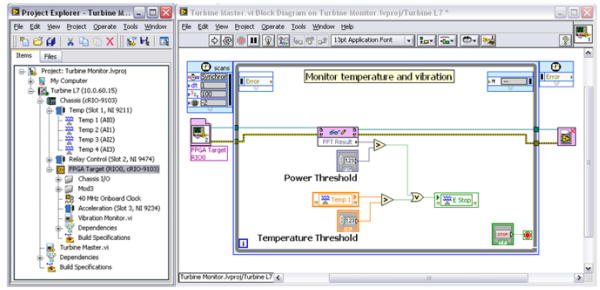
Electrical Eng.



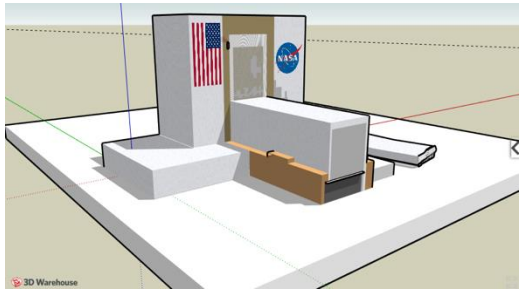

Mechanical Eng.



Control Eng.

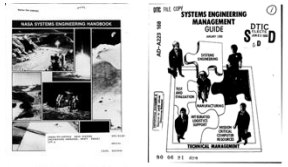


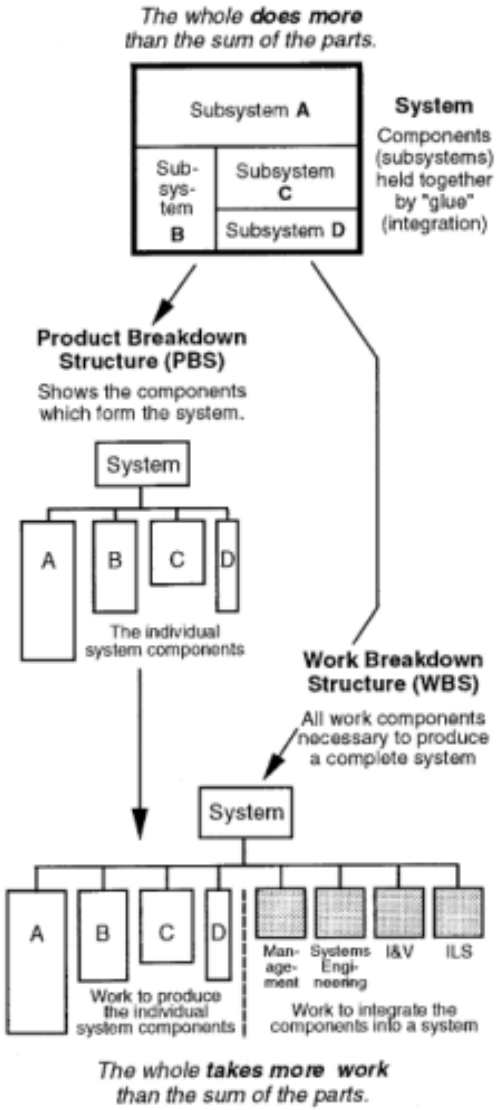

Infrastructure Eng.

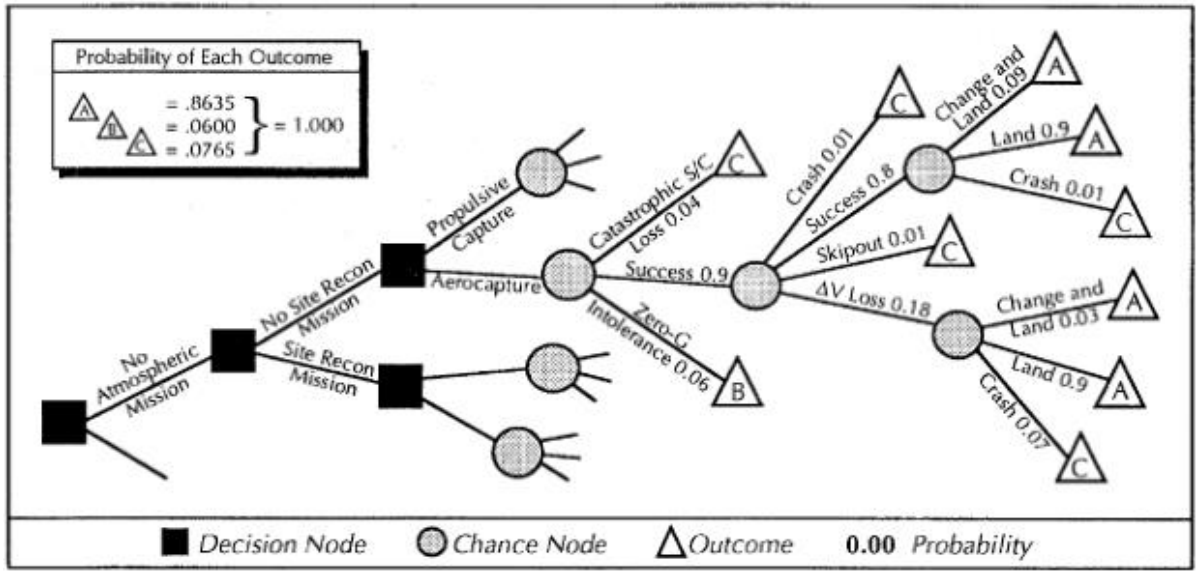


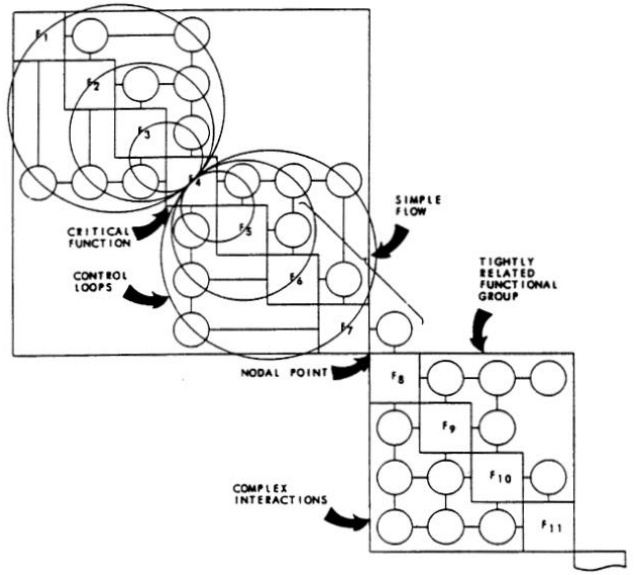

Systems Eng ???

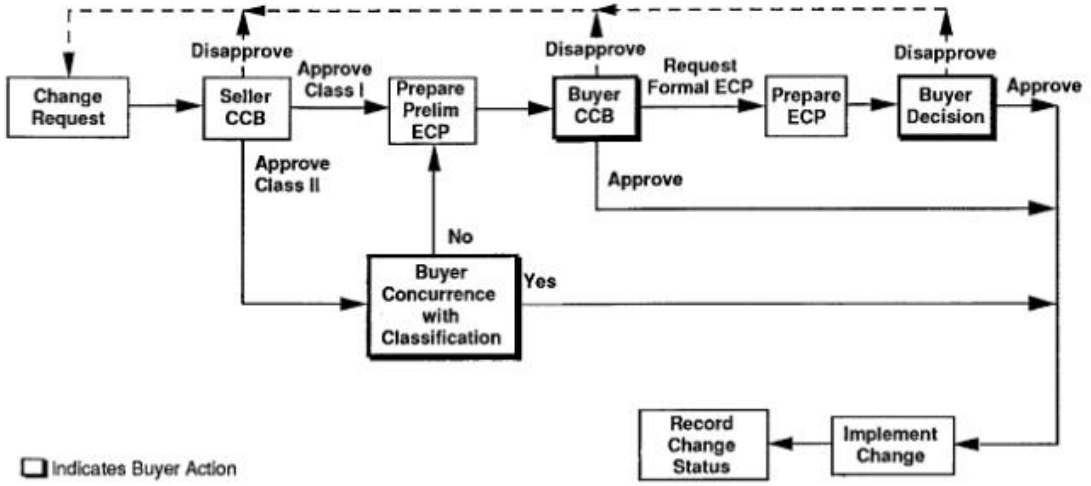
# What is Systems Engineering's language?

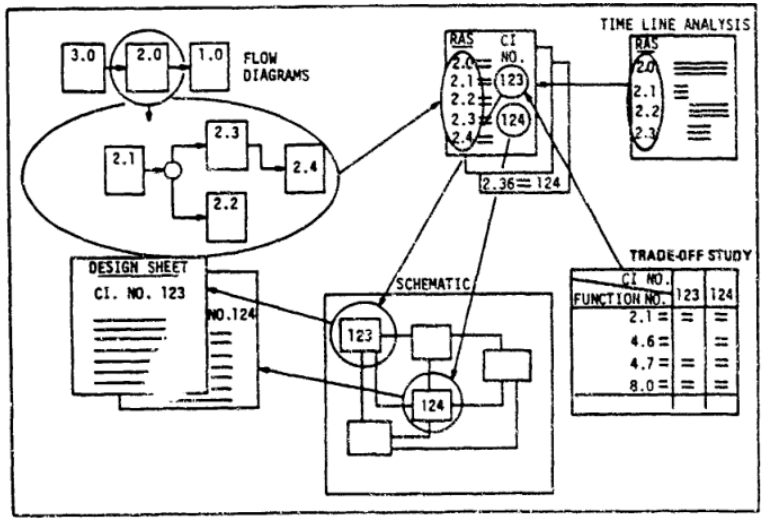




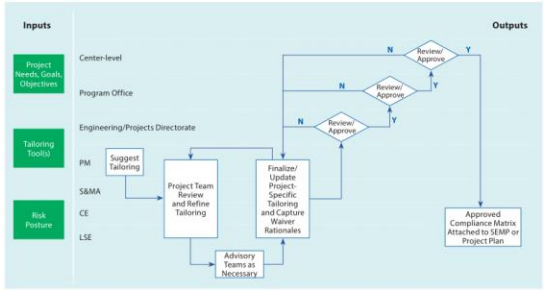



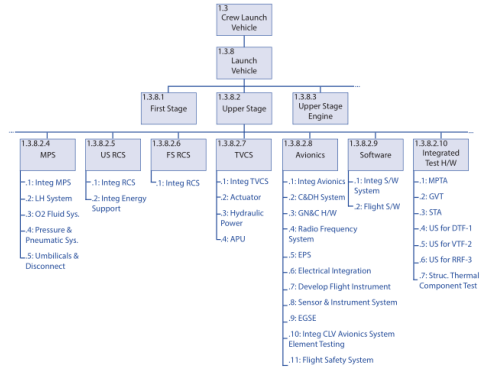



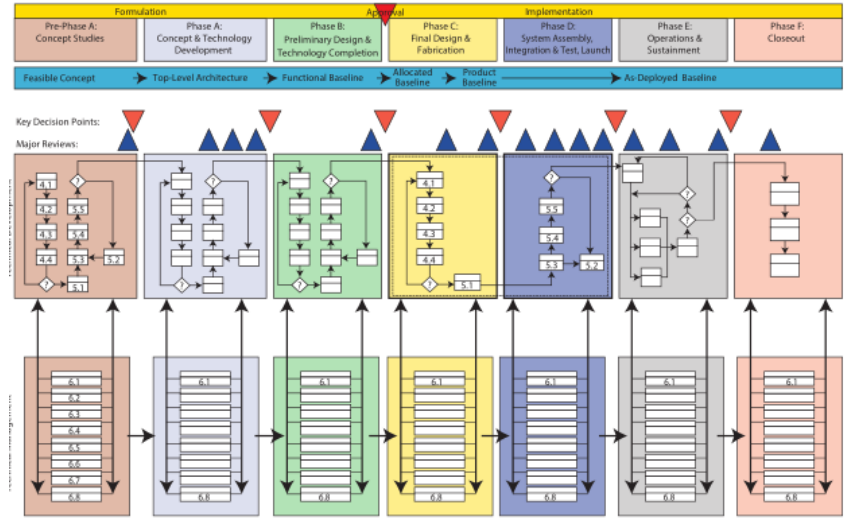



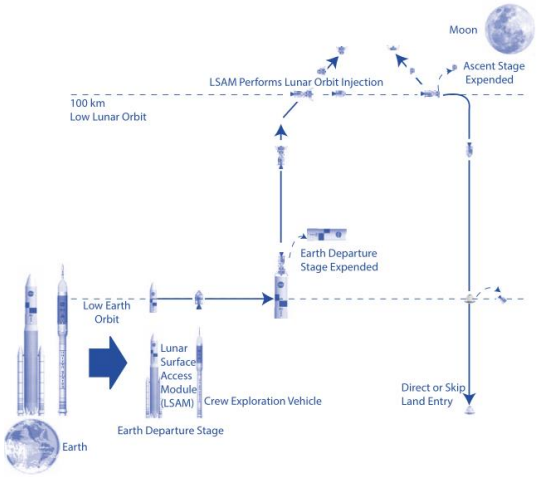



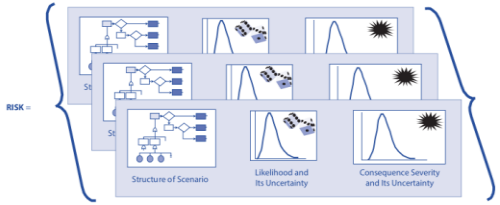
# Over time… They became colorful



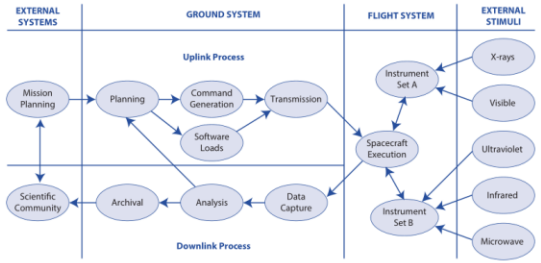



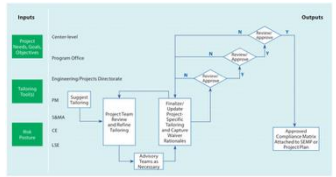

TABLE D-1 Requirements Verification Matrix

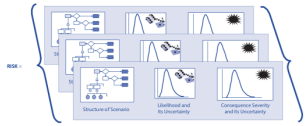
| Requirement No. | Document | Paragraph | Shall Statement | Verification Success Criteria | Verification Method | Facility or Lab | Phase* | Acceptance Required? | Preflight Acceptance? | Performing Organization | Results |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Unique identifier or each requirement* | *Document number the requirement is contained within* | *Paragraph number of the requirement* | *Text (within reason) of the requirement, i.e., the "shall"* | *Success criteria for the requirement* | *Verification method for the requirement (analysis, inspection, demonstration, test)* | *Facility or laboratory used to perform the verification and validation.* | *Phase in which the verification and validation will be performed.* | *Indicate whether this requirement is also verified during initial acceptance testing of each unit.* | *Indicate whether this requirement is also verified during any pre-flight or recurring acceptance testing of each unit* | *Organization responsible for performing the verification* | *Indicate documents that contain the objective evidence that requirement was satisfied* |
| P-1 | xxx | 3.2.1.1 Capability: Support Uplinked Data (LDR) | System X shall provide a max. ground-to-station uplink of… | 1. System X locks to forward link at the min and max data rate tolerances<br>2. System X locks to the forward link at the min and max operating frequency tolerances | Test | xxx | 5 | Yes | No | xxx | TPS xxxx |
| P-i | xxx | Other paragraphs | Other "shalls" in PTRS | Other criteria | xxx | xxx | xxx | Yes/No | Yes/No | xxx | Memo xxx |
| S-i or other unique designator | xxxxx (other specs, ICDs, etc.) | Other paragraphs | Other "shalls" in specs, ICDs, etc. | Other criteria | xxx | xxx | xxx | Yes/No | Yes/No | xxx | Report xxx |

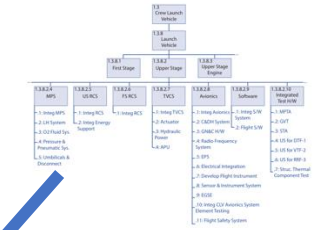
Red = Below TRL 3
Yellow = TRL 3, 4 & 5
Green = TRL 6 and above
White = Unknown
X = Exists

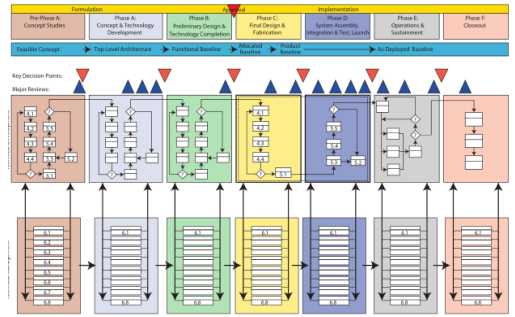
| | Concept | Demonstration Units: Breadboard | Demonstration Units: Brassboard | Demonstration Units: Developmental Model | Demonstration Units: Prototype | Demonstration Units: Flight Qualified | Environment: Laboratory Environment | Environment: Relevant Environment | Environment: Space Environment | Environment: Space Launch Operation | Unit Description: Form | Unit Description: Fit | Unit Description: Function | Unit Description: Appropriatae Scale | Overall TRL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.0 System | | | | | | | | | | | | | | | |
| 1.1 Subsystem X | | | | | | | | | | | | | | | |
| 1.1.1 Mechanical Components | | | | | | | | | | | | | | | |
| 1.1.2 Mechanical Systems | | | | | | | | | | | | | | | |
| 1.1.3 Electrical Components | | | | | X | | | | X | | X | X | X | | |
| 1.1.4 Electrical Systems | | | | | | | | | | | | | | | |
| 1.1.5 Control Systems | | | | | | | | | | | | | | | |
| 1.1.6 Thermal Systems | | | | | | | X | | | | X | X | | | |
| 1.1.7 Fluid Systems | | X | | | | | | | | | | | | | |
| 1.1.8 Optical Systems | | | | | | | | | | | | | | | |
| 1.1.9 Electro-Optical Systems | | | | | | | | | | | | | | | |
| 1.1.10 Software Systems | | | | | | | | | | | | | | | |
| 1.1.11 Mechanisms | X | | | | | | | | | | | | | | |
| 1.1.12 Integration | | | | | | | | | | | | | | | |
| 1.2 Subsystem Y | | | | | | | | | | | | | | | |
| 1.2.1 Mechanical Components | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

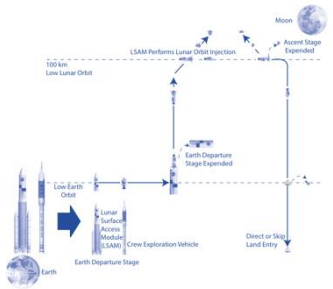
# The Systems Engineering Team is the integrator

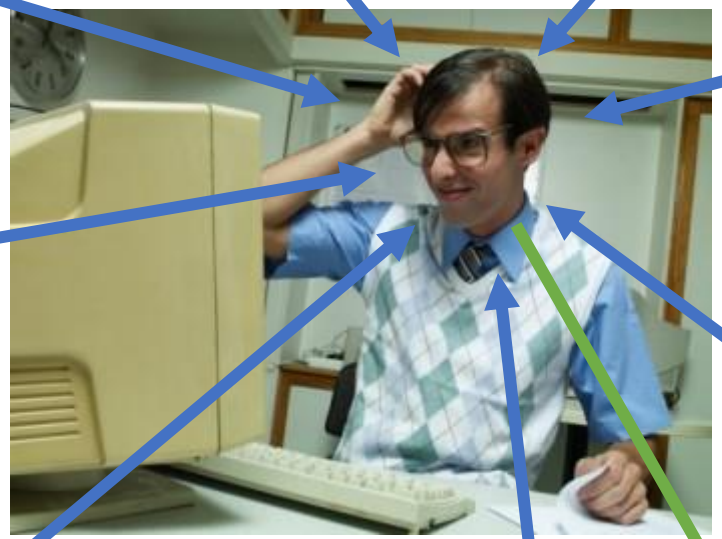


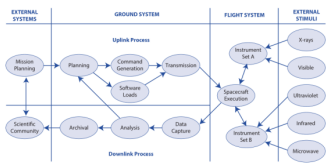


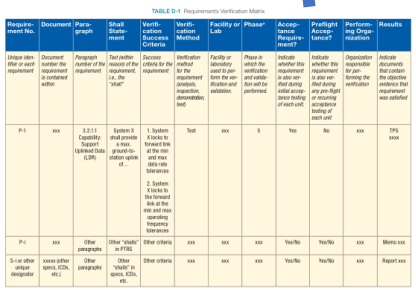


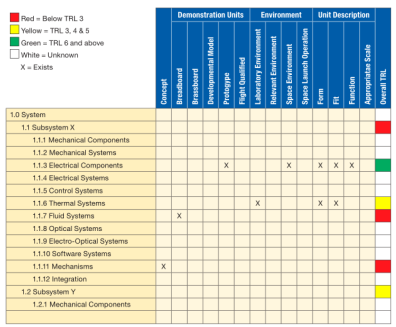




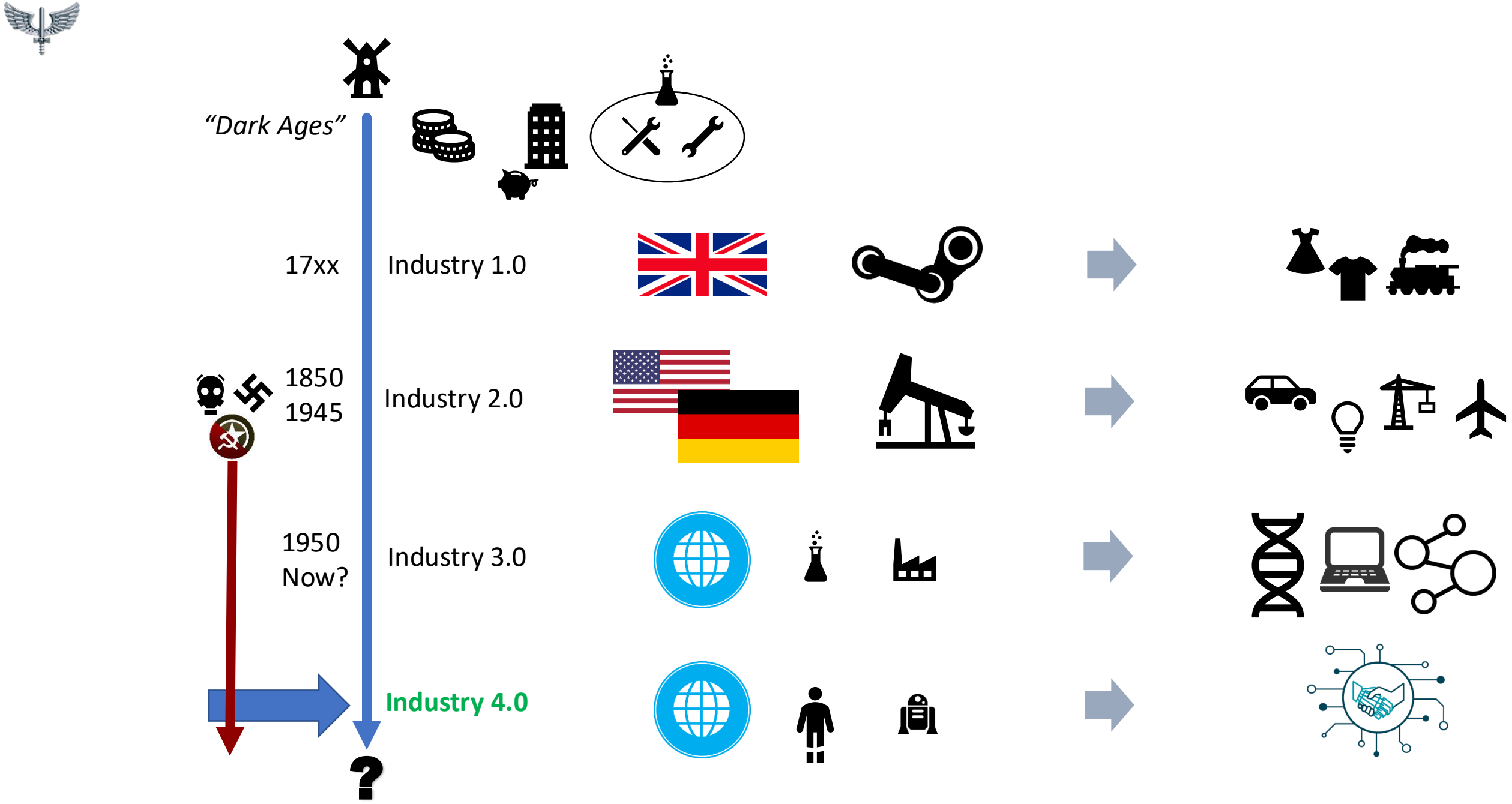


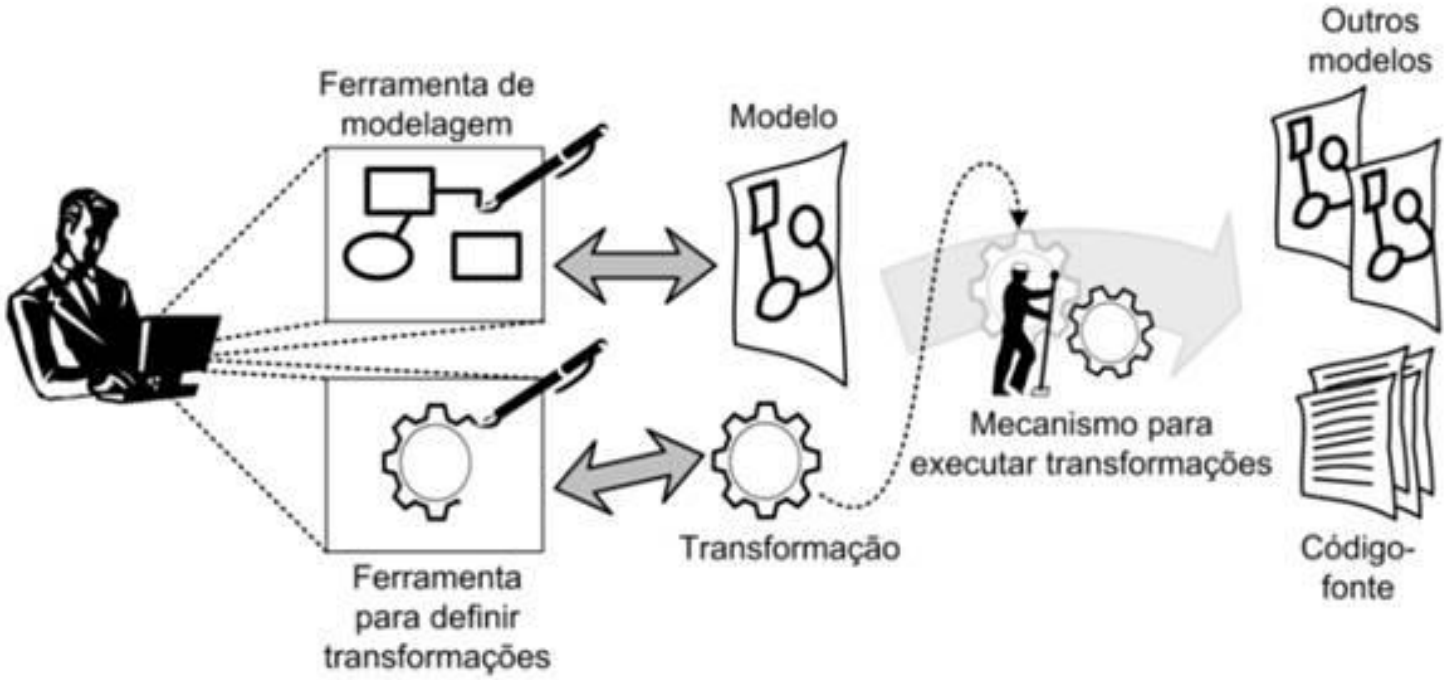



Integrator

# Digitalization of the Activities


"Dark Ages"
17xx
Industry 1.0
1850
1945
Industry 2.0
1950
Now?
Industry 3.0
Industry 4.0
?

# Model Driven Development ...




- **MDD** – Model Driven Development
- **MDSD** – Model Driven Software Development
- **MDA** – Model Driven Architect
- **MDSE** – Model Driven Software Engineering
- **MDRE** – Model Driven Reverse Engineering
- **MM** – Model Management
- **ADM** – Architecture Driven Modernization
- **DDD** – Domain Driven Design
- **MBD** – Model Based Development

http://www.devmedia.com.br/model-driven-architecture-com-enterprise-architect/33898

# Model Based Systems Engineering

# MBSE

**“Model-based systems engineering (MBSE)** is a systems engineering methodology that focuses on creating and exploiting domain models as the primary means of information exchange between engineers, rather than on document-based information exchange.”

“the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend **toward model-centric approaches** adopted by other engineering disciplines, including mechanical, electrical and software. In particular, MBSE is expected to **replace the document-centric** approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes.”

http://www.omgwiki.org/MBSE/doku.php

# So… MBSE requires a methodology… Why?

- **The tool will not provide an universal language**, it filter the "universe" of symbols into a set of options.

- **Methodology** - Defined as a collection of related processes, methods, and tools.
  - **Process** - A logical sequence of tasks performed to achieve a particular objective. A process defines the "**WHAT**" is to be done, without specifying the "HOW" each task is to be performed.
  - **Method** - Consists of techniques for performing a task, the "**HOW**" of each task. The terms "method," "technique," practice," and "procedure" can be used interchangeably in this context.
  - **Tool** - An instrument that, when applied to a particular method, can enhance the efficiency of a task. Thus, methods help bridge the gap between process and tools. The purpose of the tool should be to facilitate the accomplishment of the "HOWs".

# INCOSE METHODOLOGIES

## List of Methodologies and Methods

### Methodologies Surveyed in INCOSE 2008 Report

| Name | Primary Point of Contact |
|---|---|
| INCOSE Object-Oriented Systems Engineering Method (OOSEM) | safriedenthal@gmail.com |
| IBM Rational Telelogic Harmony-SE | peter.hoffmann@telelogic.com |
| IBM Rational Unified Process for Systems Engineering (RUP-SE) | mcantor@us.ibm.com |
| Vitech Model-Based Systems Engineering (MBSE) Methodology\|Vitech | jlong@vitechcorp.com |
| JPL State Analysis (SA) Methodology\|JPL State Analysis (SA) | Robert.D.Rasmussen@jpl.nasa.gov |
| Dori Object-Process Methodology (OPM) | dori@ie.technion.ac.il |

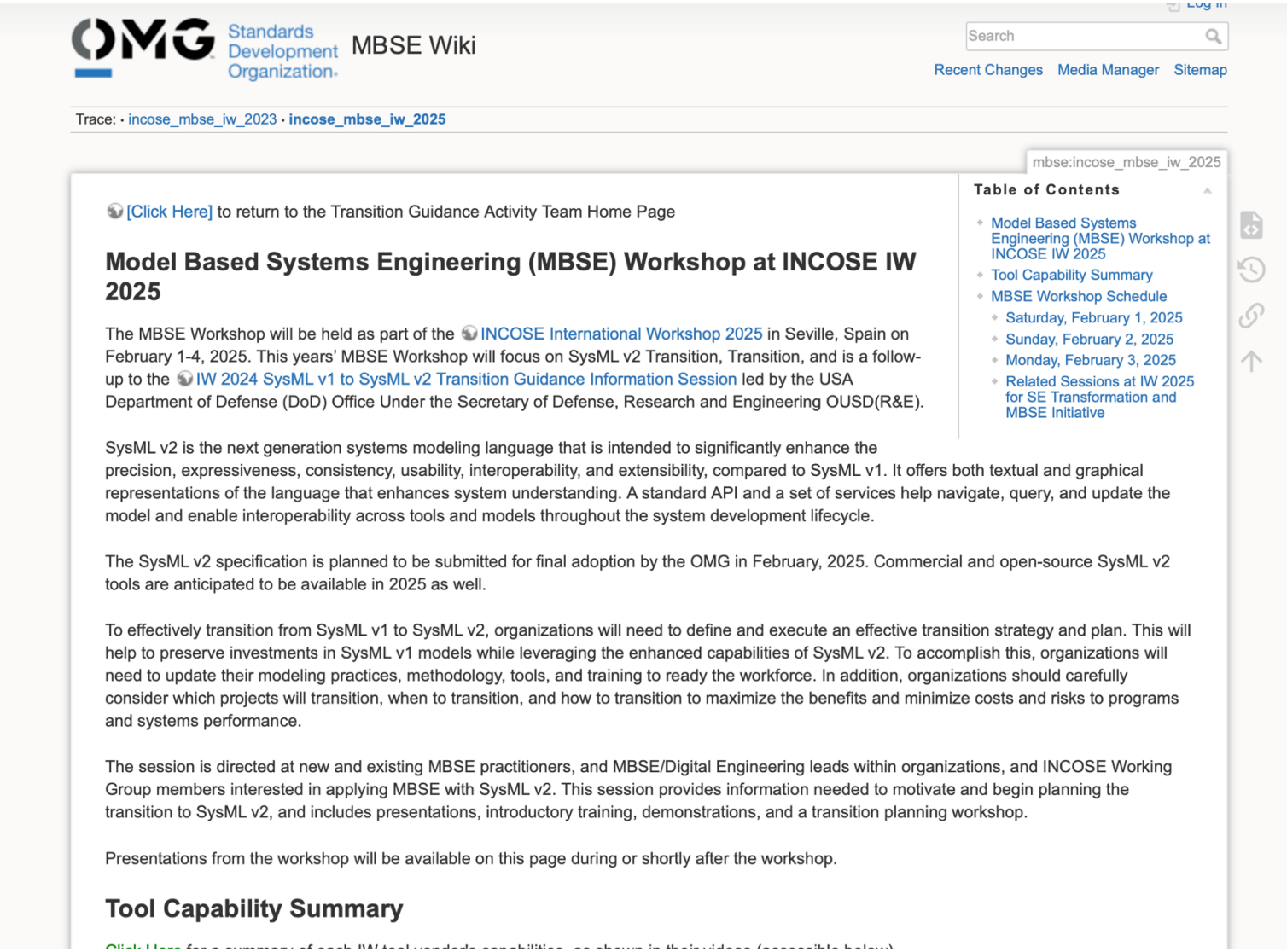
### Additional Methodologies Identified as Gaps Since 2008 INCOSE Survey

| Name | Primary Point of Contact |
|---|---|
| Weilkiens Systems Modeling Process (SYSMOD) | Tim.Weilkiens@oose.de |
| Fernandez Process Pipelines in OO Architectures (PPOOA) | joselfernandez@telefonica.net |
| An Ontology for State Analysis: Formalizing the Mapping to SysML | nicolas.f.rouquette@jpl.nasa.gov |
| ISO-15288, OOSEM and Model-Based Submarine Design | Paul.Pearce@deepbluetech.com.au |
| Alstom ASAP methodology | marco.ferrogalini@transport.alstom.com |
| Pattern-Based Systems Engineering (PBSE) | schindel@ictt.com |
| Arcadia, a model-based engineering method | Polarsys/Capella |



https://www.omgwiki.org/MBSE/doku.php?id=mbse:methodology

# Workgroup for MBSE



OMG Standards Development Organization MBSE Wiki

Search

Recent Changes Media Manager Sitemap

Trace: • incose_mbse_iw_2023 • incose_mbse_iw_2025

mbse:incose_mbse_iw_2025

Table of Contents

- Model Based Systems Engineering (MBSE) Workshop at INCOSE IW 2025
- Tool Capability Summary
- MBSE Workshop Schedule
  - Saturday, February 1, 2025
  - Sunday, February 2, 2025
  - Monday, February 3, 2025
  - Related Sessions at IW 2025 for SE Transformation and MBSE Initiative

[Click Here] to return to the Transition Guidance Activity Team Home Page

## Model Based Systems Engineering (MBSE) Workshop at INCOSE IW 2025

The MBSE Workshop will be held as part of the INCOSE International Workshop 2025 in Seville, Spain on February 1-4, 2025. This years' MBSE Workshop will focus on SysML v2 Transition, Transition, and is a follow-up to the IW 2024 SysML v1 to SysML v2 Transition Guidance Information Session led by the USA Department of Defense (DoD) Office Under the Secretary of Defense, Research and Engineering OUSD(R&E).

SysML v2 is the next generation systems modeling language that is intended to significantly enhance the precision, expressiveness, consistency, usability, interoperability, and extensibility, compared to SysML v1. It offers both textual and graphical representations of the language that enhances system understanding. A standard API and a set of services help navigate, query, and update the model and enable interoperability across tools and models throughout the system development lifecycle.

The SysML v2 specification is planned to be submitted for final adoption by the OMG in February, 2025. Commercial and open-source SysML v2 tools are anticipated to be available in 2025 as well.

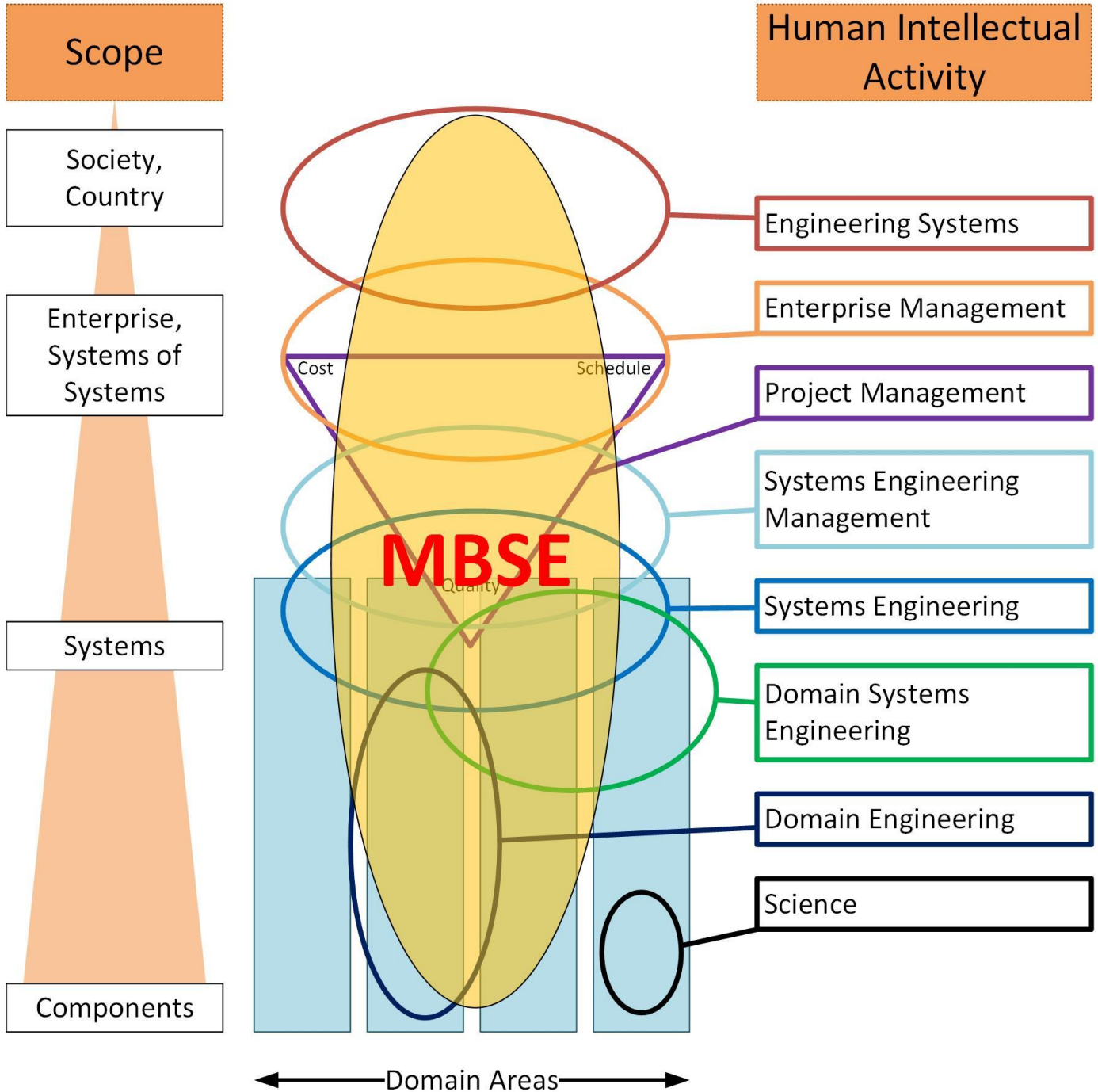
To effectively transition from SysML v1 to SysML v2, organizations will need to define and execute an effective transition strategy and plan. This will help to preserve investments in SysML v1 models while leveraging the enhanced capabilities of SysML v2. To accomplish this, organizations will need to update their modeling practices, methodology, tools, and training to ready the workforce. In addition, organizations should carefully consider which projects will transition, when to transition, and how to transition to maximize the benefits and minimize costs and risks to programs and systems performance.

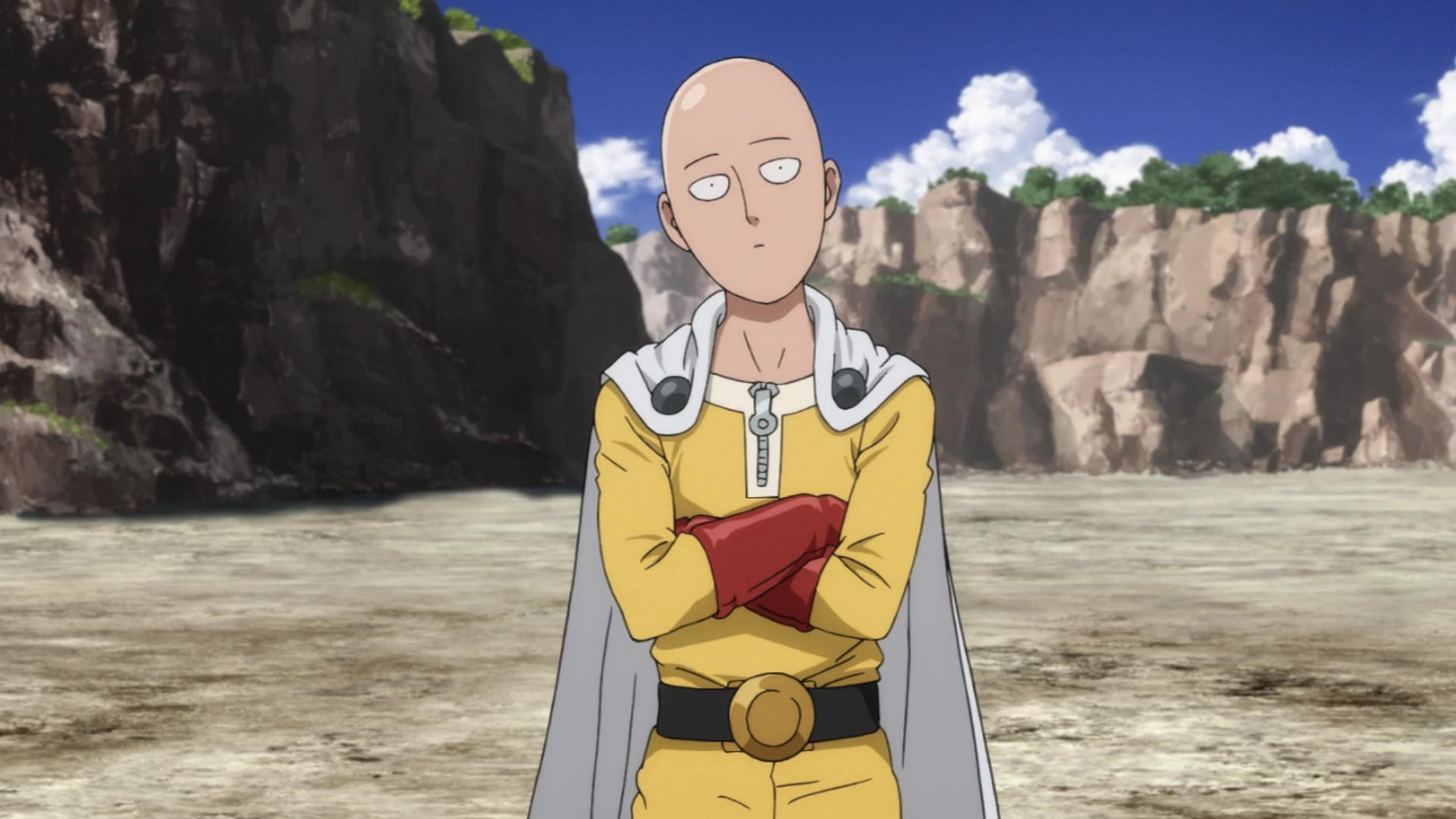
The session is directed at new and existing MBSE practitioners, and MBSE/Digital Engineering leads within organizations, and INCOSE Working Group members interested in applying MBSE with SysML v2. This session provides information needed to motivate and begin planning the transition to SysML v2, and includes presentations, introductory training, demonstrations, and a transition planning workshop.

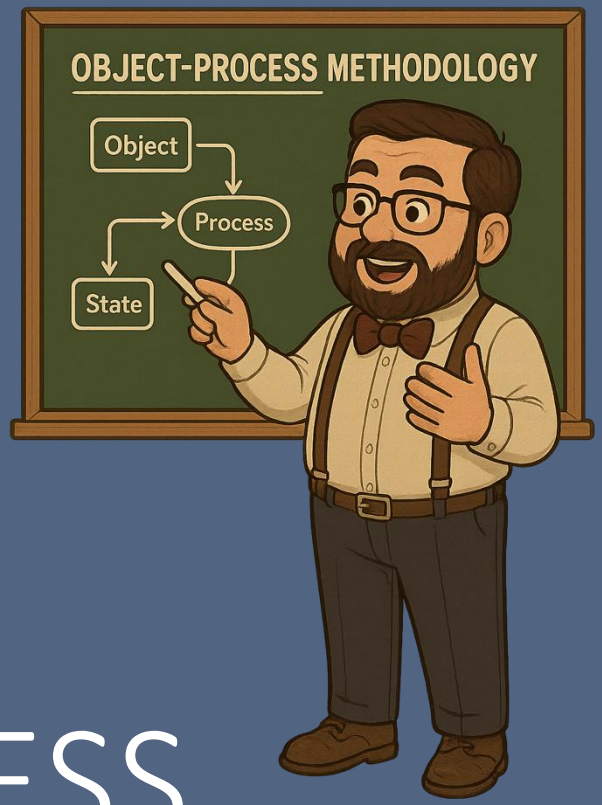
Presentations from the workshop will be available on this page during or shortly after the workshop.

### Tool Capability Summary

https://www.omgwiki.org/MBSE/doku.php?id=mbse:incose_mbse_iw_2025


Scope
Human Intellectual Activity
Society, Country
Enterprise, Systems of Systems
Systems
Components
Cost
Schedule
Quality
MBSE
Engineering Systems
Enterprise Management
Project Management
Systems Engineering Management
Systems Engineering
Domain Systems Engineering
Domain Engineering
Science
Domain Areas






# OPM – OBJECT PROCESS METHODOLOGY

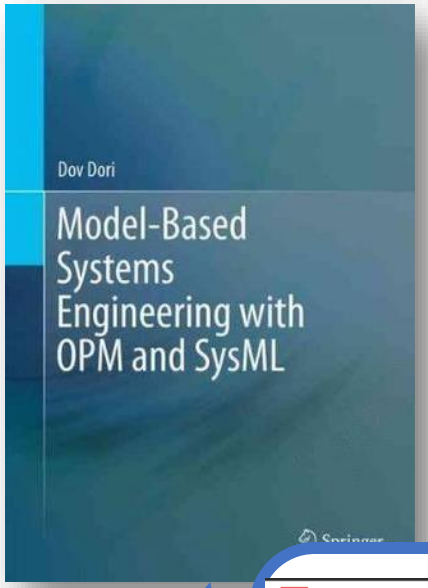
2025 – Prof. Dr. Christopher Shneider Cerqueira

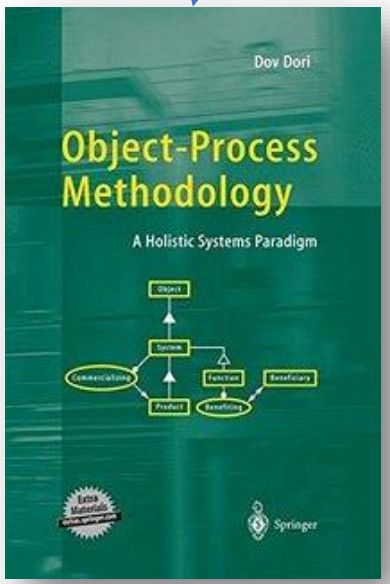
# OPM

Prof. Dov Dori

Created in 2002

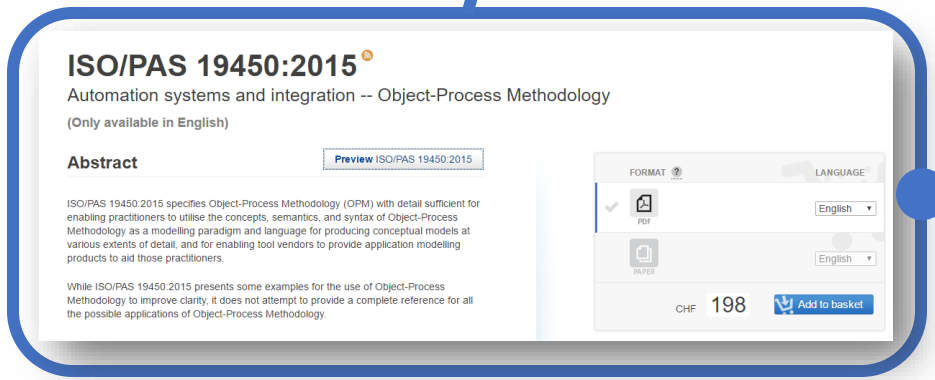
- **A** Single Diagram – Maps Behaviour and Structure
- **2 Building Blocks and 10 basic relations**
- Designed to “Systemic View” and “Concept Modelling”
- Simulation Ready

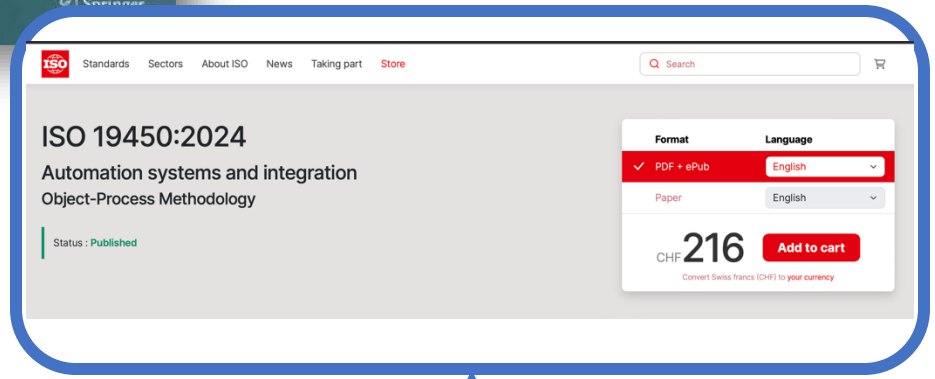


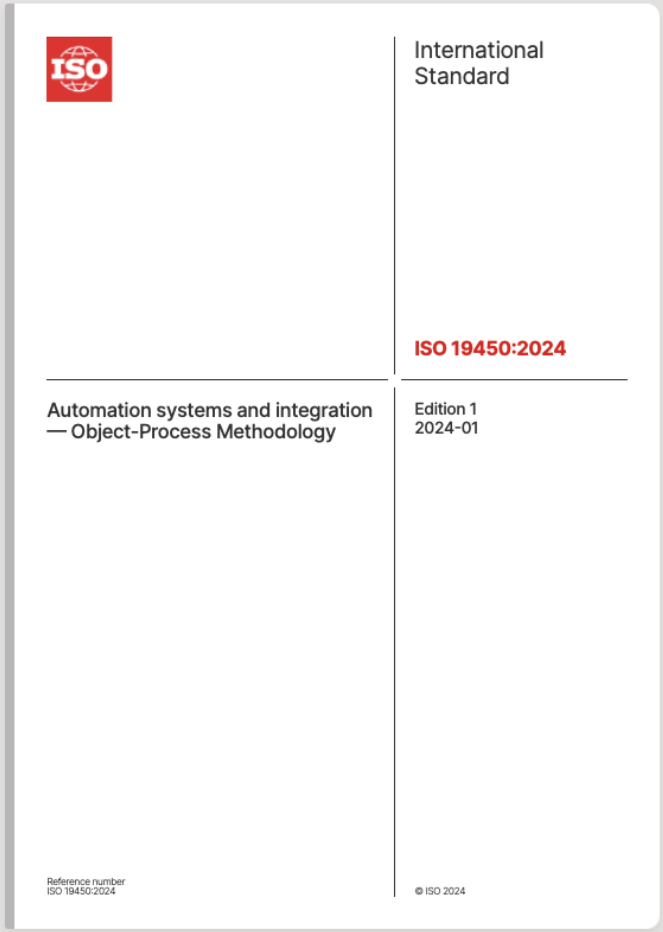

improving and showing it applicability

- ~130 Pages standard
- Published in late 2015
- Intended to *“Automation Systems and Integration”*
- Has the “**power**” of a ISO seal.




Added complexity and integrations




Read sample

# ISO 19450:2024

## Automation systems and integration — Object-Process Methodology

**Published** (Edition 1, 2024)

# I had the honor to met Prof. Dori in 2022


# References for this section

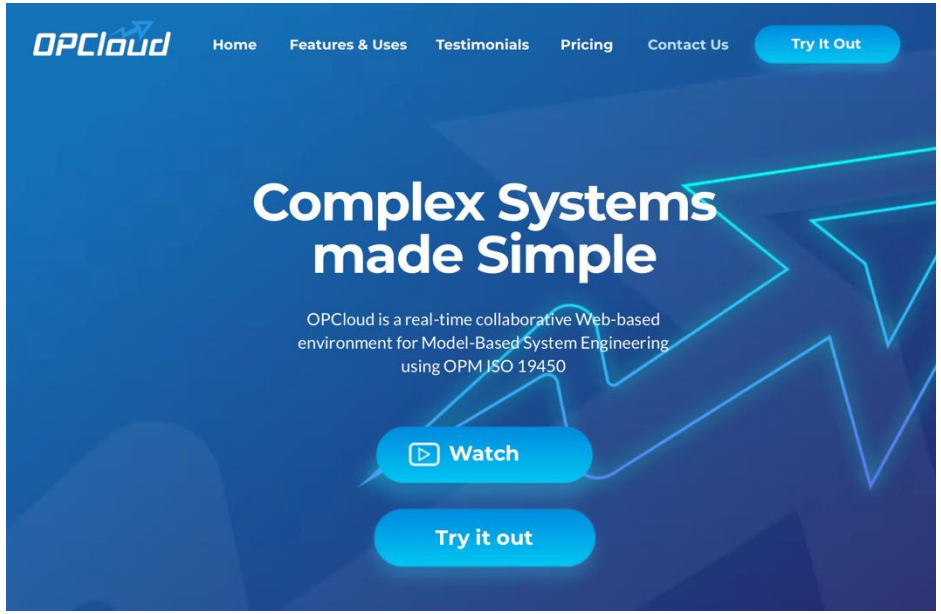
- Main:
  - DORI, D. Model-Based Systems Engineering with OPM and SysML. New York: Springer, 2016. ISBN 978-1-4939-3294-8.
- OPM Starting Guide
  - http://esml.iem.technion.ac.il/introduction-to-opm/
- Absurdly good:
  - https://en.wikipedia.org/wiki/Object_Process_Methodology

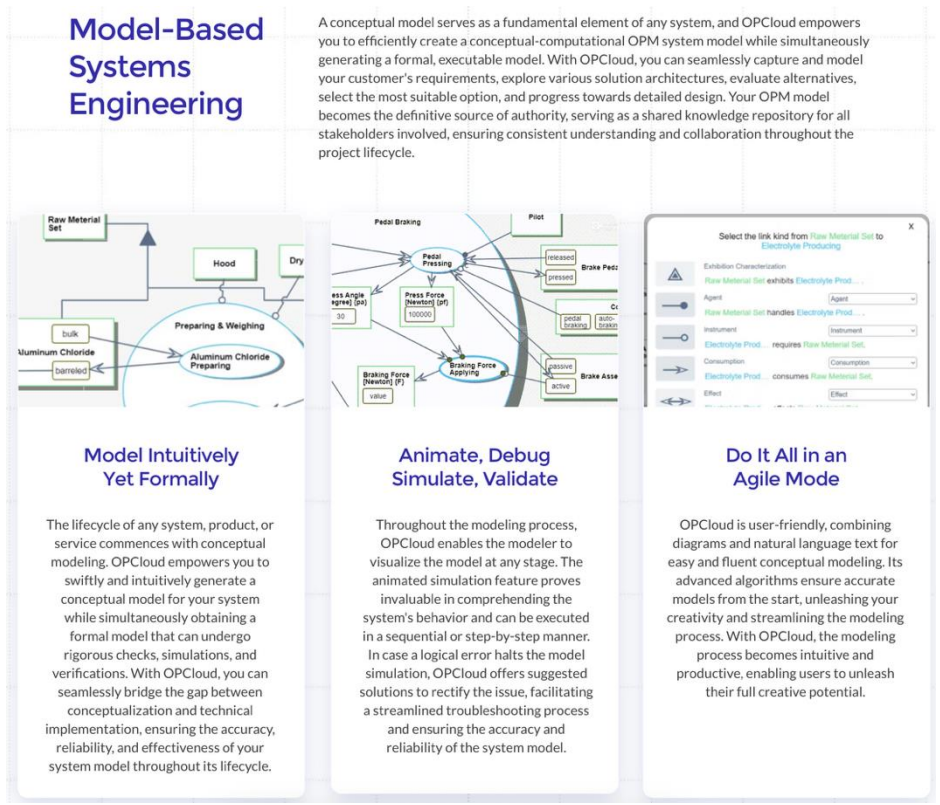


The following slides have “pieces” of the presentations/papers of the OPM Starting Guide

# Resources for this lecture

- OPCloud
- **Logged account:**
  - https://opcloud.systems/
  - Connect w/ Google SSO using your **@ga account**
- **Not logged**
  - It **does not save** it, but works similarly
  - https://www.opcloud.tech



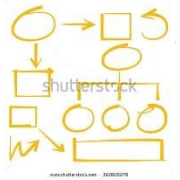



# Conceptual Modeling

# The start: How do we explain ideas to each other?

Grab a pen and piece of paper, or a chalk and blackboard

Scribble shapes with names next to them




While talking, run lines with or without arrows among the shapes

Follow the reaction of the audience to see if idea is understood

Answer questions, continue scribbling…

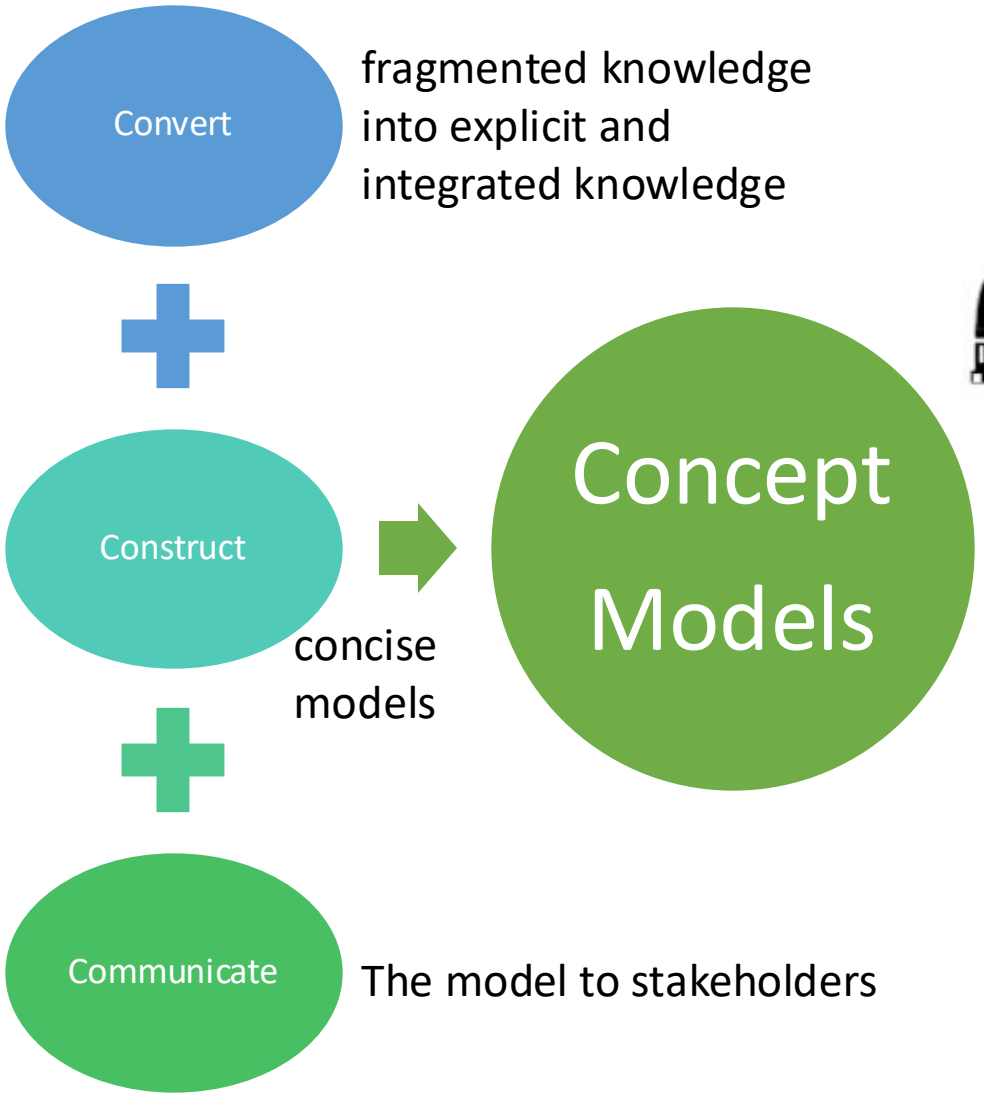
# The start: These “first” ideas → Conceptual Modeling

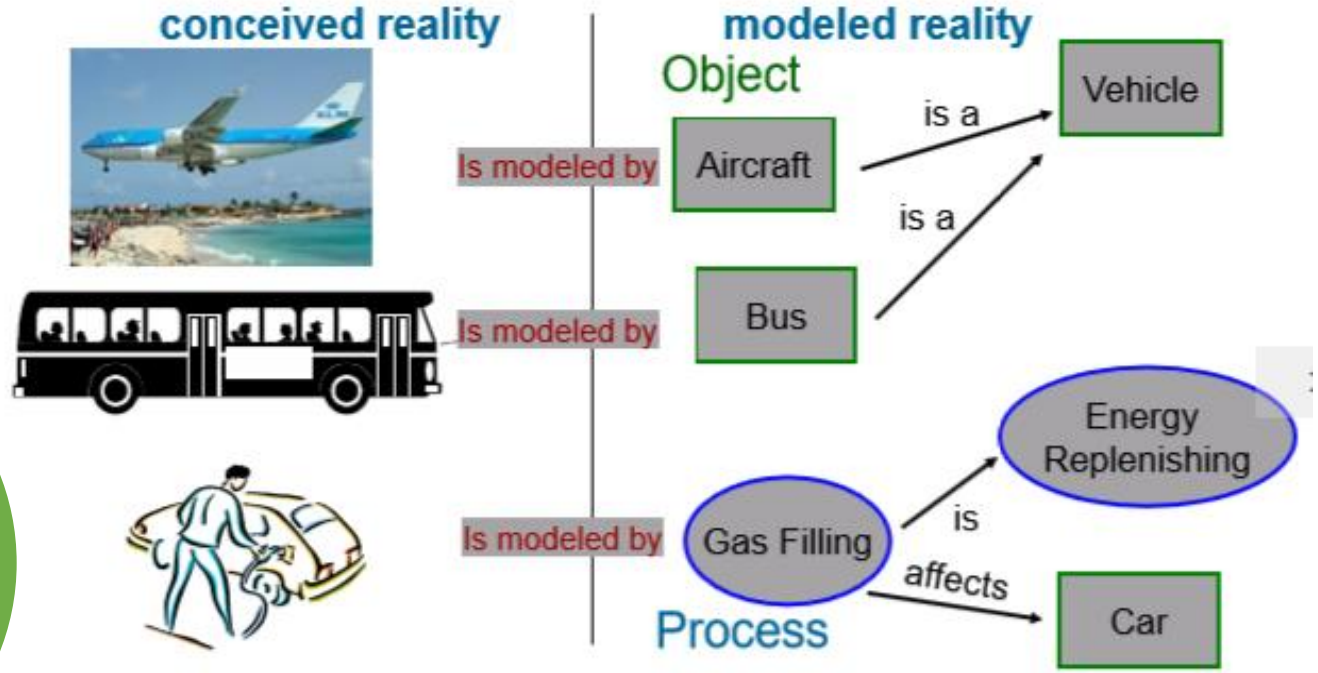
- A systematic, formalized process of describing, specifying, designing or explaining ideas, systems, products or processes through a model

- Applicable to both:
  - **Science** – Studying what is known and what is missing to satisfy human thirst for knowledge, and
  - **Engineering** – Designing systems to benefit humans, based on sound scientific principles

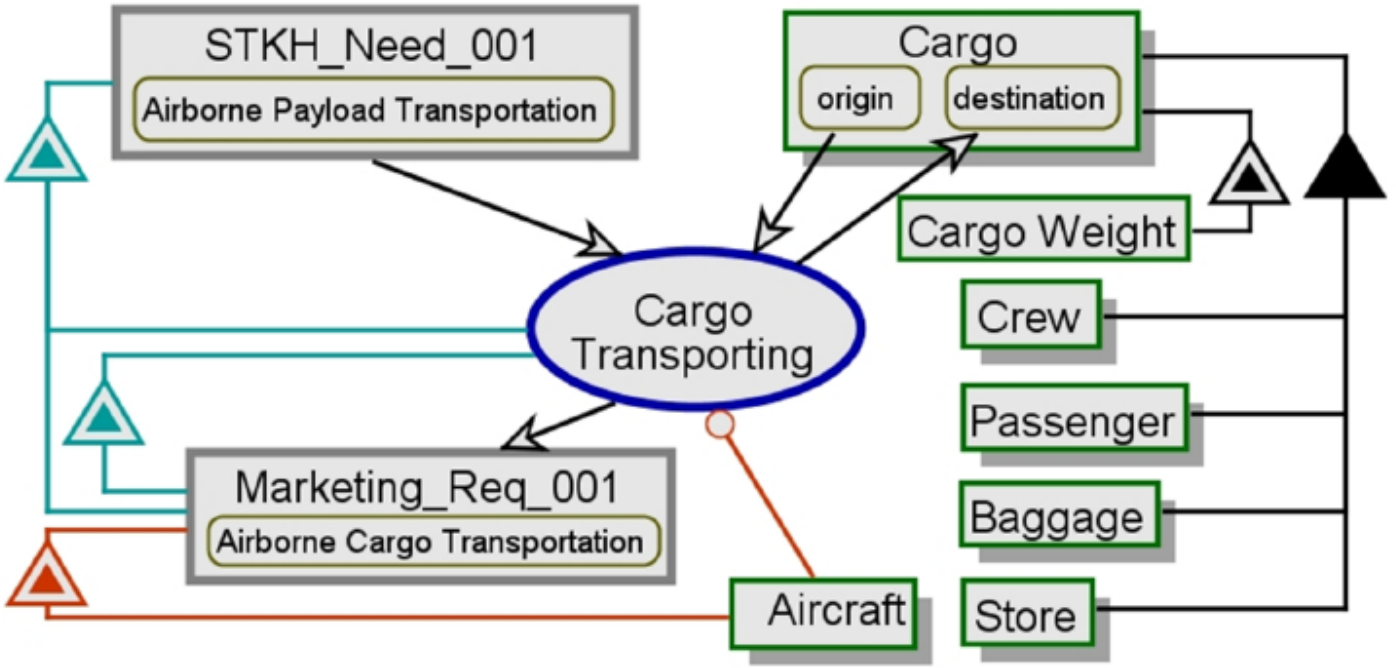
- Science can be thought of as reverse engineering of nature

# Conceptual Modelling




- **simple** yet **expressive**, and
- **intuitive** yet **formal**




STKH_Need_001 is Airborne Payload Transportation.
Cargo Transporting consumes STKH_Need_001.
Cargo Transporting yields Marketing_Req_001.
Marketing_Req_001 is Airborne Cargo Transportation.
Marketing_Req_001 exhibits Aircraft.
Cargo Transporting requires Aircraft.
Aircraft is physical.
Cargo Transporting changes Cargo from origin to destination.

Function Defining
Requirements Identifying
Requirements Allocating

STKH_Need_001 exhibits Marketing_Req_001, as well as Cargo Transporting.
Cargo Transporting exhibits Marketing_Req_001.

Traceability

Cargo is physical.
Cargo can be origin or destination.
Cargo exhibits Cargo Weight.
Cargo consists of Passenger, Baggage, Store, and Crew.
Passenger is physical.
Baggage is physical.
Store is physical.
Crew is physical.

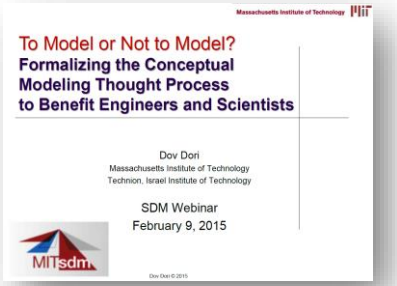
Configuration
management



**Model-Based Systems Engineering for Aircraft Design with Dynamic Landing Constraints Using Object-Process Methodology ,** AIAA Scitech 2019 Forum

# OPM Foundations

# [Dori] Fundamental Questions:


To Model or Not to Model?
Formalizing the Conceptual Modeling Thought Process to Benefit Engineers and Scientists
Dov Dori
Massachusetts Institute of Technology
Technion, Israel Institute of Technology
SDM Webinar
February 9, 2015
MITsdm

- **1. What is needed to describe the universe?**
  - "Things" and their "relations"
- **2. What can those things do?**
  - Things can exist or happen.
- **3. What are the things that exist in the world?**
  - Objects exist – statics (time-independent).
- **4. What are the things that happen in the world?**
  - Processes happen – are dynamics (time-dependent).

# [Dori] Fundamental Questions:

- **5. How do objects and processes relate?**
  - Processes happen to objects. While happening,
  - Processes transform objets.
- **6. Transform?? what does that mean?**
  - Create
  - Destroy
  - Affect
    - an object

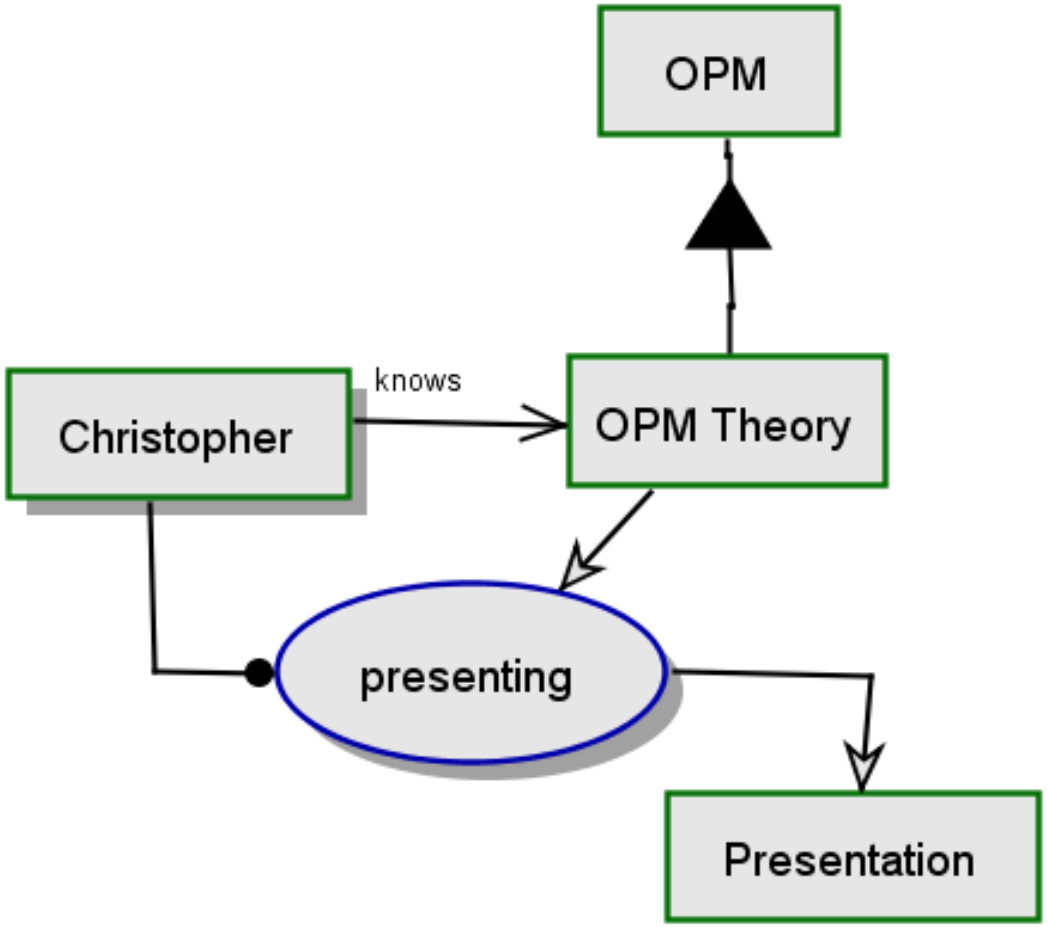
# [Dori] Fundamental Questions:

- **7. Affecting? What does that mean?**
  - A process affects an object by changing its state. Hence, objects must have states.
- **8. What are the two major aspects of any system?**
  - Structure: static aspect – what the system is made of?
  - Behaviour: dynamic aspect - how the system changes over time?
- **9. Which third aspect is specific to man-made systems?**
  - Function: the utilitarian, subjective aspect. Why? for whom? Who benefits?

# [Dori] Object-Process Theorem

**Objects** with **states**, **processes** and their relations among them constitute a **necessary** and **sufficient universal ontology** to describe a **system**.

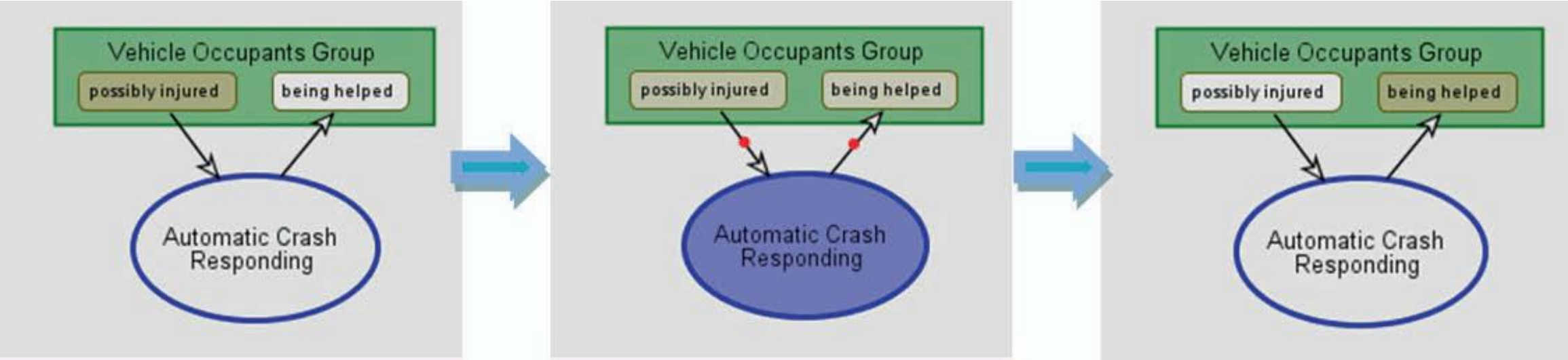
# Cognitive channels: visual-OPD and textual-OPL

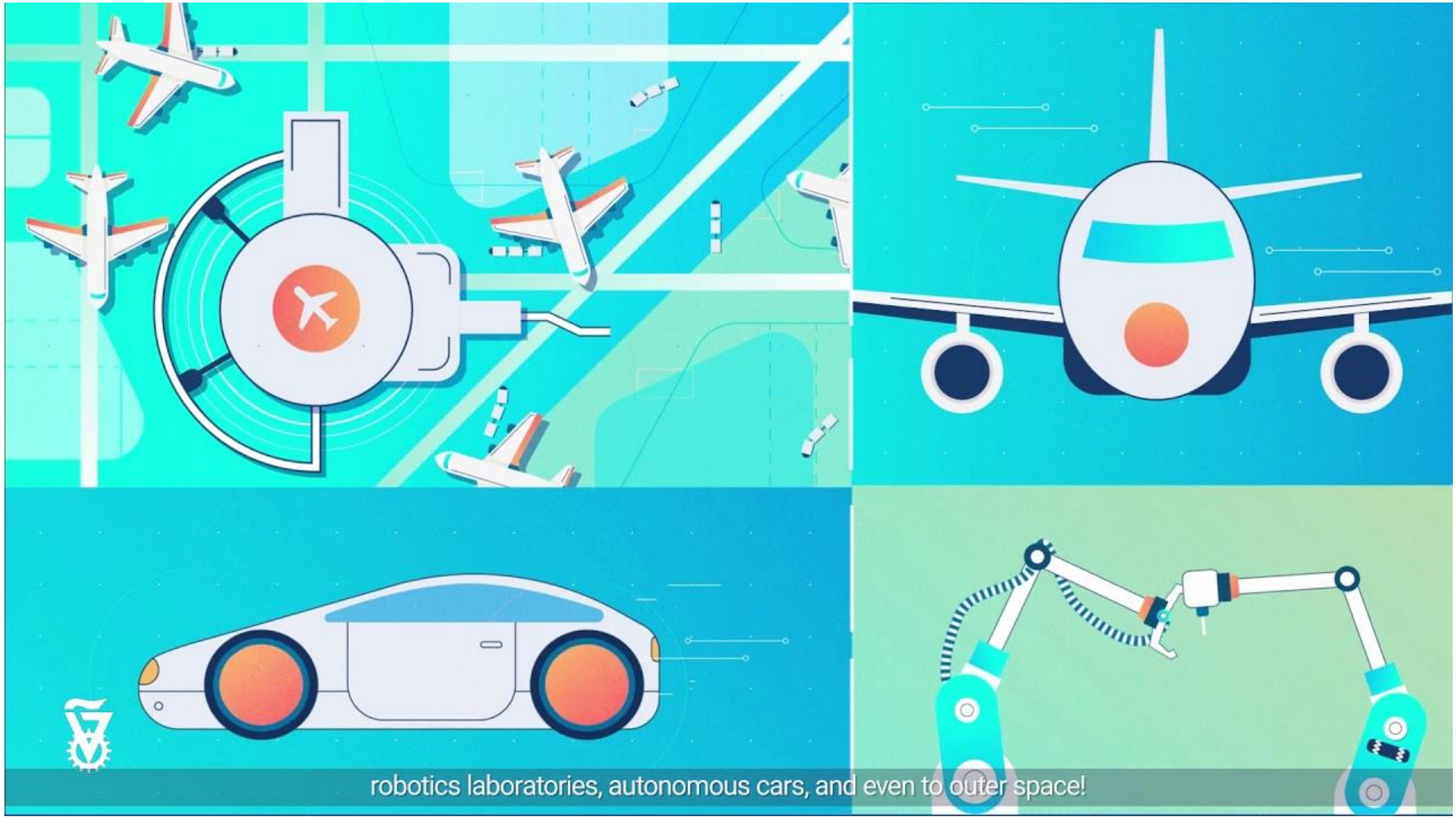



Christopher is physical.
Christopher knows OPM Theory.
Christopher handles presenting.
OPM consists of OPM Theory.
presenting is physical.
presenting consumes OPM Theory.
presenting yields Presentation.

# Model Simulation

- One of the most attractive and useful features of an OPM model, which enables it to be visualized and tested, is its executability; that is, the ability to simulate a system by executing its model via animation in a properly designed software environment.



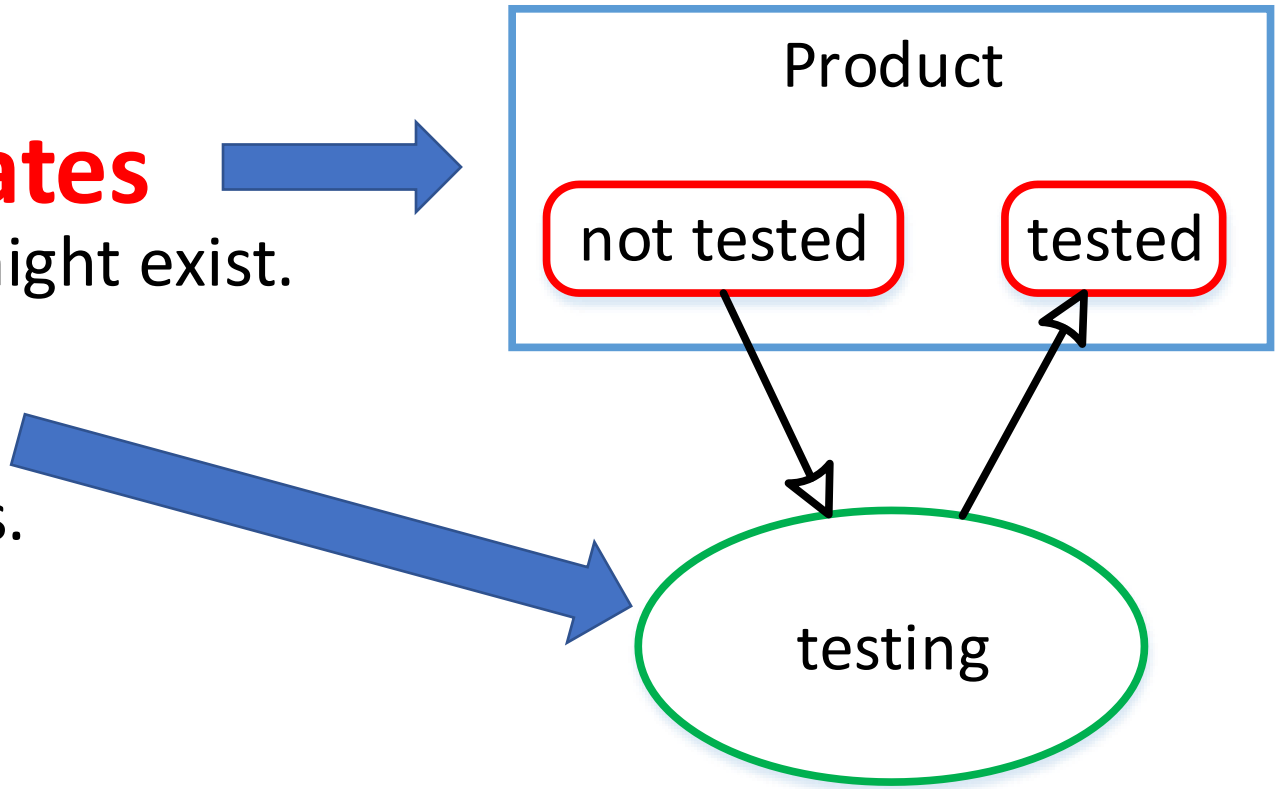



https://www.opcloud.tech/

# Building Blocks

# OPM has only two building blocks:

1. **Objects with states**
   A thing that exists or might exist.
2. **Processes**
   A thing that transforms.



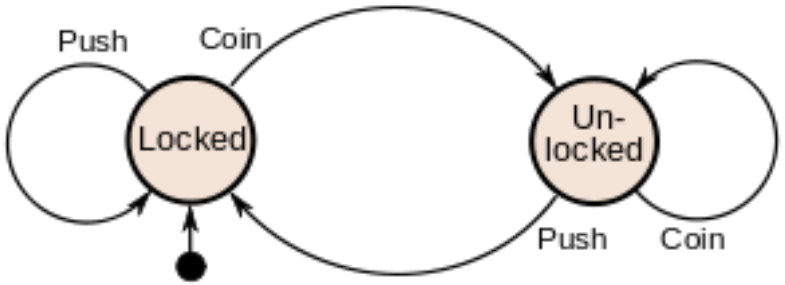

All the other elements are **relations** between things, expressed graphically as links

# A Object Thing

- An **object** is a **thing** that exists. Its existence can be either physical or conceptual. An **object** is a **thing** that can be transformed.
- An **object** can represent simple things such as car keys, or complex systems such as manufacturing plants.
- The graphical representation of an **object** in
- OPM is a square:

# States and Values

- A **state** is a possible situation at which an object can be.
- A **state** is only meaningful in within the context of a containing object.
- The graphical representation of a **state** in OPM is a rountangle (rounded rectangle):




# A Process Thing

- A **process** is a thing that transforms an **object**.
- This transformation can be:
  - Creation of an **object**.
  - Consumption of an **object**.
  - Changing the state of an **object**.
- By definition, a process must be associated with at least one object.
- The graphical representation of a process in OPM is an ellipse:

# Things nature

- Things
  - **Existential** essence
  - Design concern of the system or outside the **boundary**

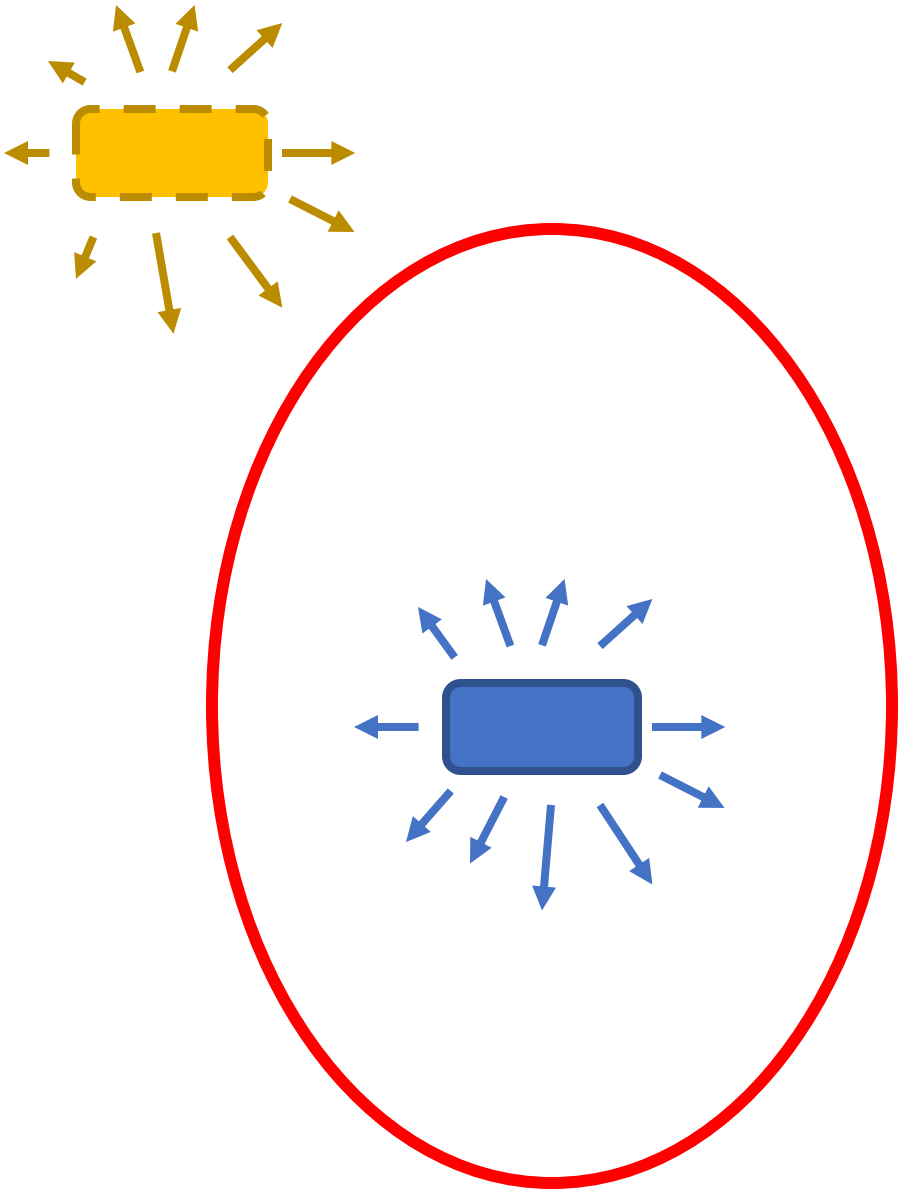
# Essence

- **Key to modeling Cyber-Physical Systems**
  - **Physical** objects in the model represent what is really “out there” –actual states and values of objects
  - **Informatical** objects represent information about their corresponding physical objects

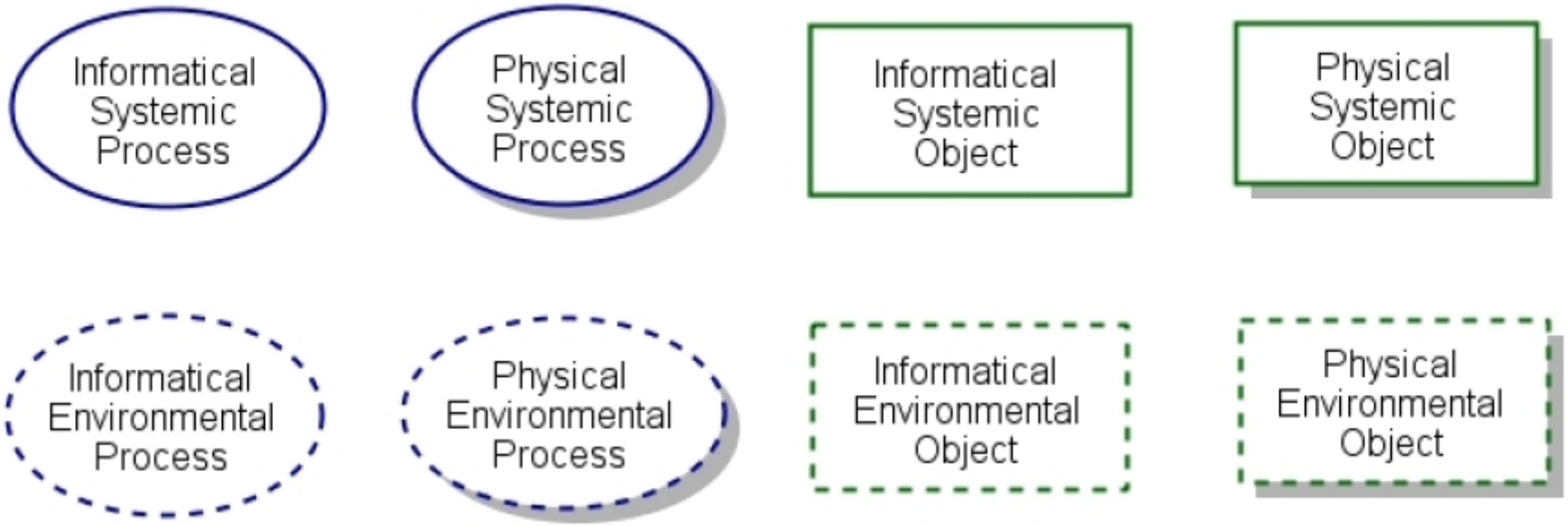
- Only ***informatical objects*** are available to a decision-making agent (human or artificial)

# Affiliation

- **Affiliation**, which pertains to the thing's scope and denotes whether the thing is:
  - **systemic**, i.e. part of the system, or
  - **environmental**, i.e. part of the system's environment.



# Essence/Affiliation




**Informatical Systemic Process** is an informatical and systemic process.
**Physical Systemic Process** is a physical and systemic process.
**Informatical Systemic Object** is an informatical and systemic object.
**Physical Systemic Object** is a physical and systemic object.
**Informatical Environmental Process** is an informatical and environmental process.
**Physical Environmental Process** is a physical and environmental process.
**Informatical Environmental Object** is an informatical and environmental object.
**Physical Environmental Object** is a physical and environmental object.

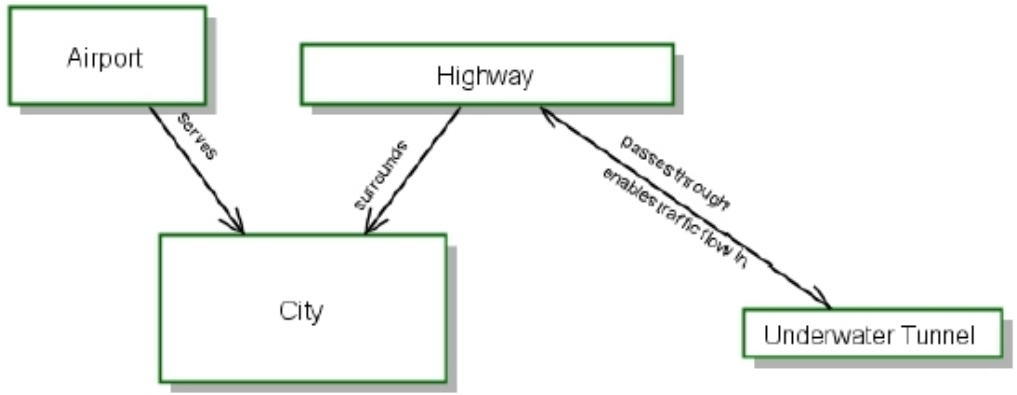
# OPM Structure

Structural Links

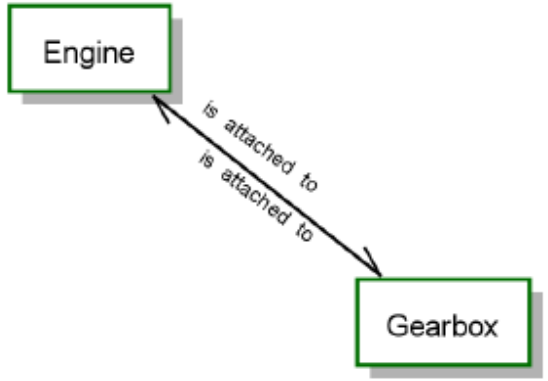
- **Structural Link** is a link that specificies **a static aspect** of the system by connecting na object to another object or a process to another process.

# Tagged Structural Links

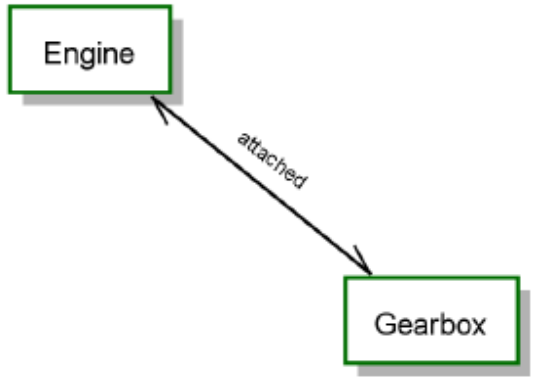
- A unidirectional tagged structural link defines a structural link between a source **object** and a **target** object.
  - The syntax of the unidirectional tagged structural link OPL sentence shall be: ***Source-thing tag Destination-thing***.
  - The syntax of the unidirectional null-tagged structural link OPL sentence shall be: ***Source-thing*** *relates to* ***Destination-thing***.
  - The syntax of the reciprocal tagged structural link with only one tag shall be: ***Source-thing*** *and* ***Destination-thing*** *are* ***reciprocity-tag***.
  - The syntax of the reciprocal tagged structural link with no tag shall be: ***Source-thing*** *and* ***Destination-thing*** *are related.*




Airport serves City.
Highway surrounds City.
Highway passes through Underwater Tunnel.
Underwater Tunnel enables traffic flow in Highway.



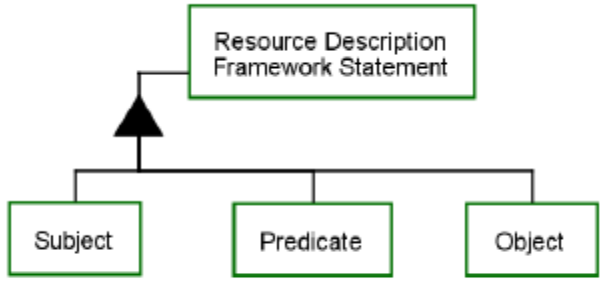

Engine is attached to Gearbox.
Gearbox is attached to Engine.



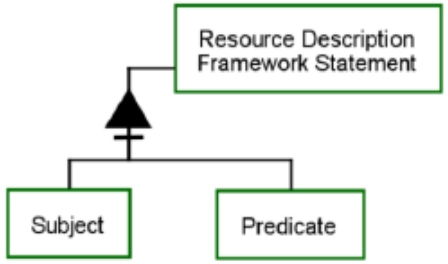

Engine and Gearbox are attached.

# Aggregation-Participation

- A structural relation which denotes that one (high level) **thing** aggregates (i.e. consists of, contains) one or more (low level) **things**. The high-level **thing** is called the *whole* or the *aggregate* while the lower-level **things** are called the *parts*.
  - The syntax of the aggregation-participation relation link shall be: ***Whole-thing*** *consists of* ***Part-thing1***, ***Part-thing2***, …, ***and Part-thingn***.




**Resource Description Framework Statement** consists of **Subject**, **Predicate**, and **Object**.




**Resource Description Framework Statement** consists of **Subject**, **Predicate**, and at least one other part.

# Exhibition-Characterization

- A structural relation which denotes that one **thing** exhibits (or is characterized) by another **thing**. A **thing** exhibits *features* that characterize it: An *attribute* is a *static feature* & An *operation* is a *dynamic feature*.
- The main difference between exhibition and aggregation is that an attribute always has a value, whether a part may be inexistent: A bag of candies will be empty after Purim (aggregation), but it will always have a color (exhibition).
  - The syntax of the exhibition-characterization relation link for an object exhibitor with a complete collection of n attributes and m operations shall be: ***Object-exhibitor exhibits Attribute1, Attribute2, … , and Attributen,*** *as well as* ***Operation1, Operator2, … , Operatorm.***
  - The syntax of the exhibition-characterization relation link for a process exhibitor with a complete collection of n operation features and m attribute features shall be: ***Process-exhibitor exhibits Operation1, Operator2, … , Operatorn,*** *as well as* ***Attribute1, Attribute2, …, and Attributem.***

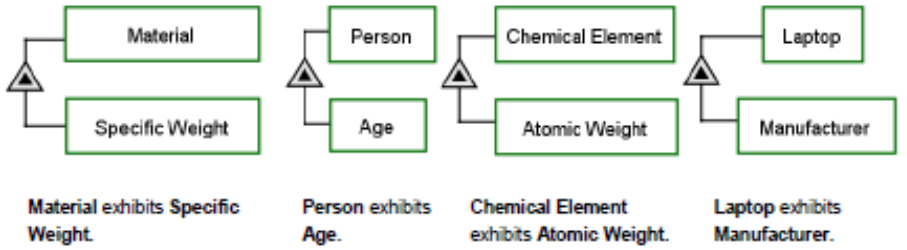



Figure 20 — Object attribute examples

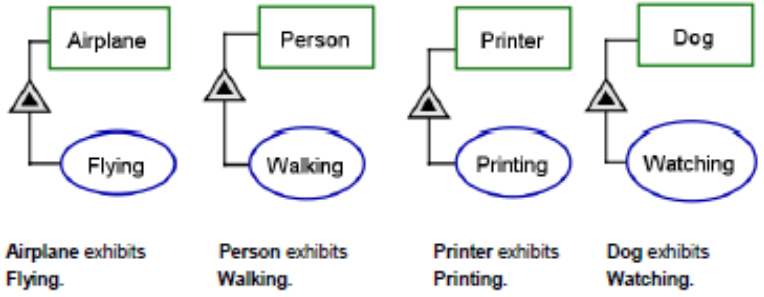



Figure 21 — Object exhibitor with operation examples

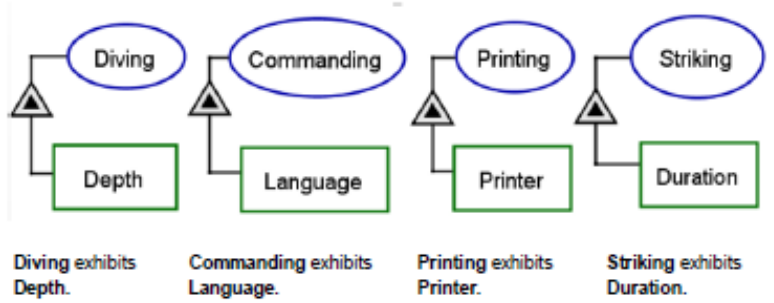
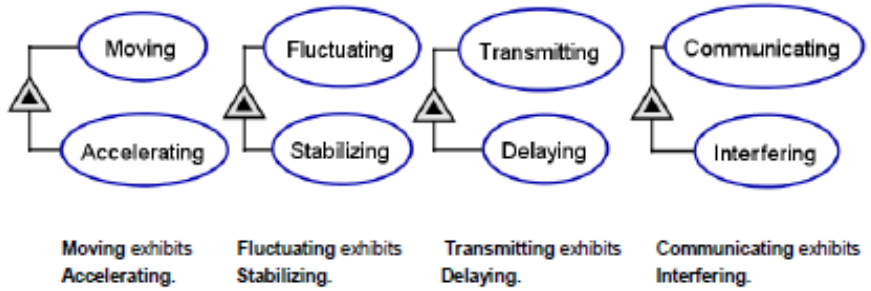
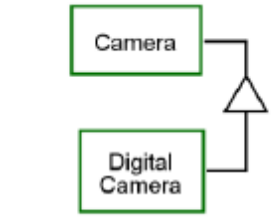



Figure 22 — Process exhibitor with attribute examples



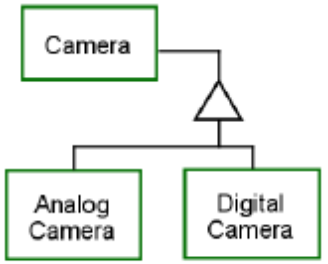
# Generalization-Specialization

- A structural relation which denotes that one **thing** specializes to another **thing**.
- Commonly referred as inheritance in OO modelling languages.
  - For a complete collection of n specializations of a general that is an object, the syntax of the generalization-specialization relation link OPL sentence shall be: ***Specialization-object1, Specialization-object2, …,*** *and* ***pecialization-objectn are General-object*****.**
  - For a complete collection of n specializations of a general that is a process, the syntax of the generalization-specialization relation link OPL sentence shall be: ***Specialization-process1, Specialization-process2, …,*** *and* ***Specialization-processn are General-process*****.**

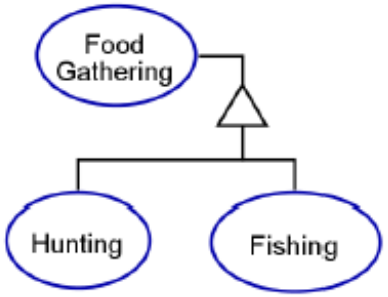



Digital Camera is a Camera.


Hunting is Food Gathering.



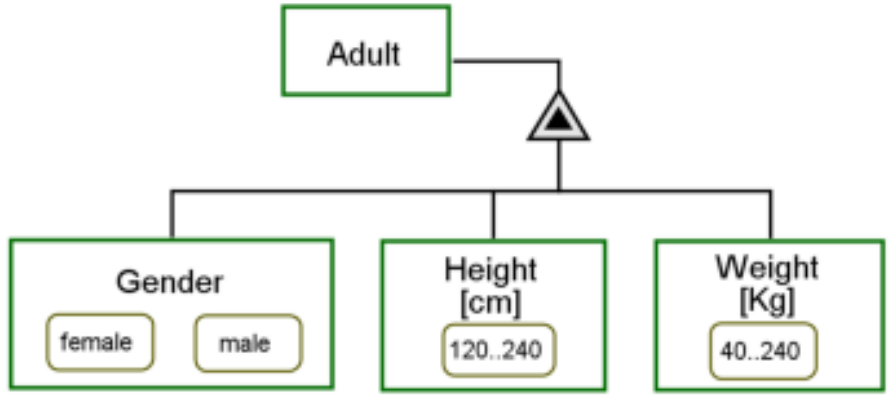

Analog Camera and Digital Camera are Cameras.



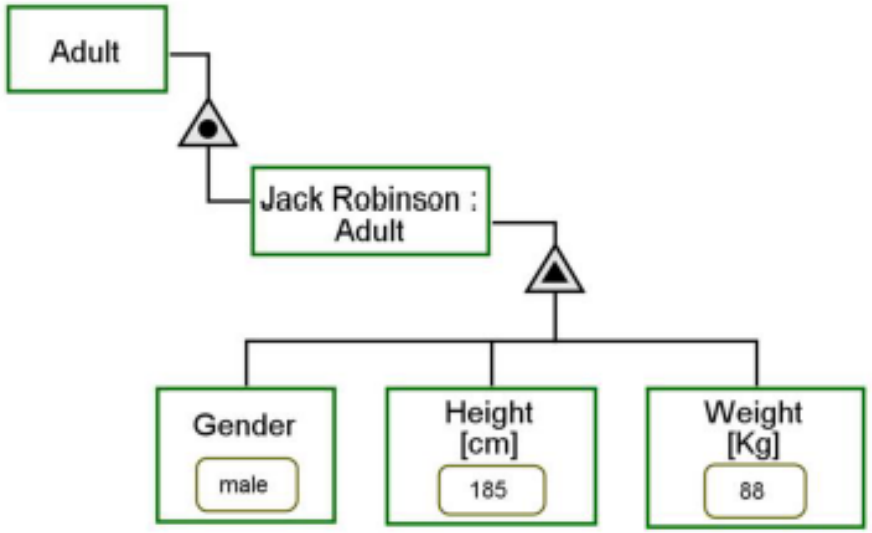

Hunting and Fishing are Food Gathering.

# Classification-Instantiation

- The classification, which is an object class or a process class, is a source pattern for a thing connecting with one or more destination things, which are instances of the source thing's pattern, i.e. the qualities the pattern specifies acquire explicit values to instantiate the instance thing.
- An instance of a class shall be an incarnation of a particular identifiable instance of that class with the same classification identifier.
  - The syntax of the classification-instantiation relation link between an object class and a single instance shall be: ***Instance-object is an instance of Class-object*.**
  - The syntax of the classification-instantiation relation link between a process class and a single instance shall be: ***Instance-process is an instance of Class-process*.**
  - The syntax of the classification-instantiation relation link between a process class and n instances shall be; ***Instance-object1, Instance-object2, …, Instance-objectn are instances of Class-object*.**
  - The syntax of the classification-instantiation relation link between a process class and n instances shall be; ***Instance-process1, Instance-process2, …, Instance-processn are instances of Class-process*.**

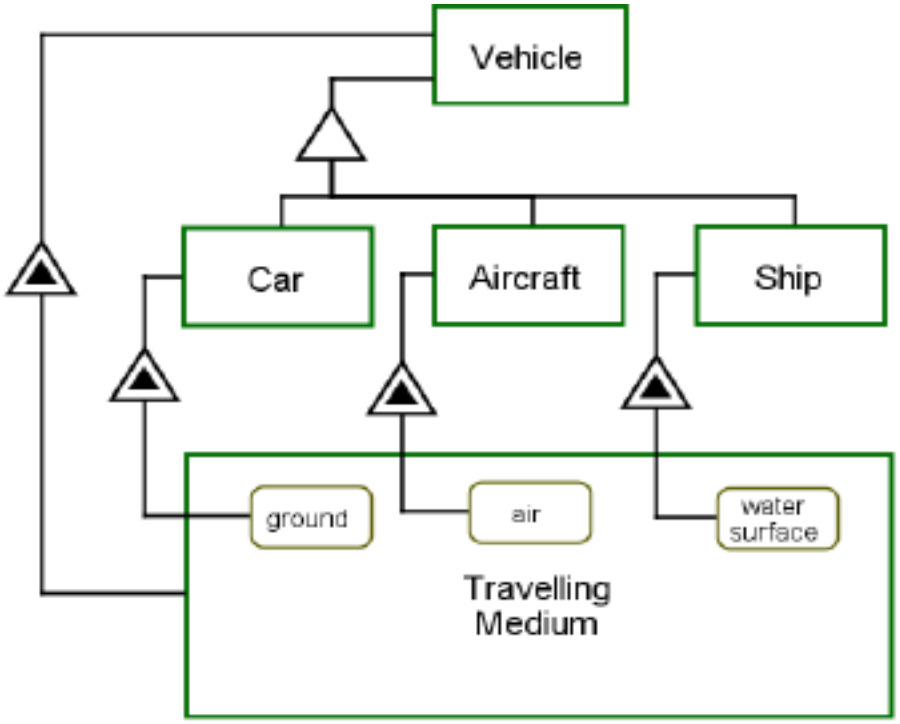



**Adult** exhibits **Gender**, **Height in cm**, and **Weight in Kg**.
**Gender** of **Adult** can be **female** or **male**.
**Height** in **cm** of **Adult** ranges from **120** to **240**.
**Weight** in **Kg** of **Adult** range from **40** to **240**.

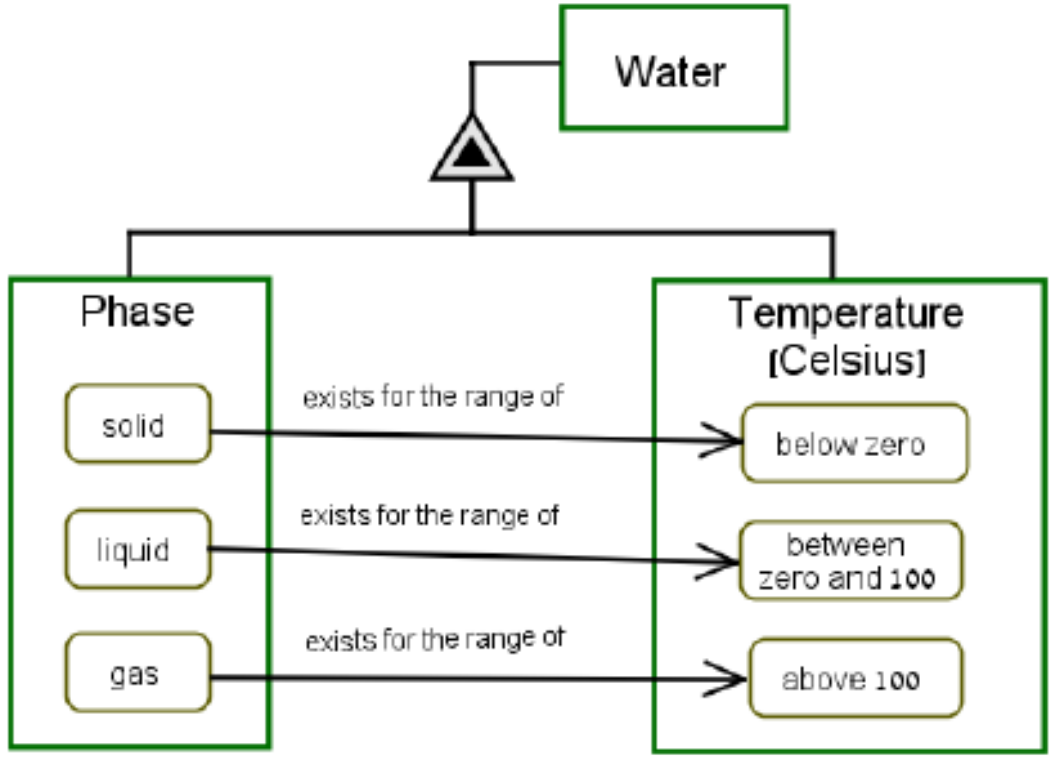



**Jack Robinson** is an instance of **Adult**.
**Gender** of **Jack Robinson** is **male**.
**Height** in **cm** of **Jack Robinson** is **185**.
**Weight** in **kg** of **Jack Robinson** is **88**.

# States can be also used in some relations

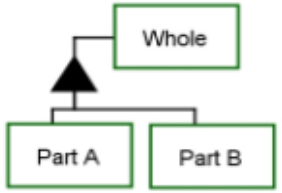
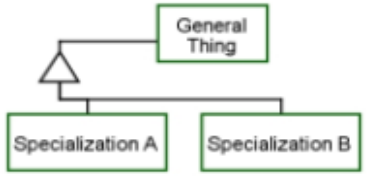
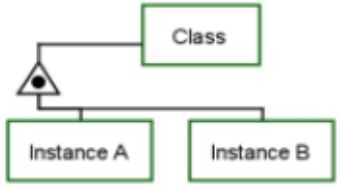
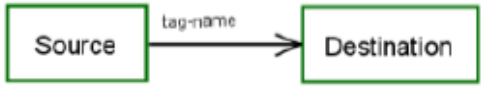
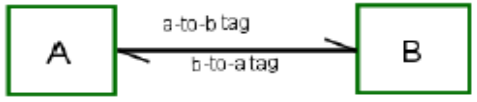
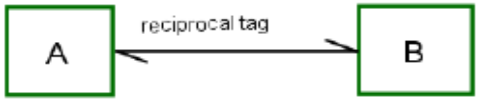



**Vehicle** exhibits **Travelling Medium.**
**Travelling Medium** of **Vehicle** can be **ground**, **air**, and **water surface.**
**Car**, **Aircraft**, and **Ship** are **Vehicles.**
**Car** exhibits **ground Travelling Medium.**
**Aircraft** exhibits **air Travelling Medium.**
**Ship** exhibits **water surface Travelling Medium.**




**Water** exhibits **Phase** and **Temperature** in **Celsius.**
**Phase** can be **solid**, **liquid**, or **gas.**
**Temperature** in **Celsius** can be **below zero**, **between zero and 100**, or **above 100.**
**Solid Phase exists for the range of below zero Temperature** in **Celsius.**
**Liquid Phase exists for the range of between zero and 100 Temperature** in **Celsius.**
**Gas Phase exists for the range of above 100 Temperature** in **Celsius.**

# Summary

| | | | |
|---|---|---|---|
| **Aggregation**-<br>Participation |  | **Whole** consists of **Part A** and **Part B**. | – |
| Exhibition-<br>**Characterization** |  | **Exhibitor** exhibits **Attribute A** as well as **Operation B**. | – |
| **Generalization**-<br>Specialization |  | – | **Specialization A** and **Specialization B** are **General Thing.** |
| **Classification**-<br>Instantiation |  | – | **Instance A** and **Instance B** are instances of **Class.** |
| Unidirectional tagged<br><br>[Unidirectional null tagged] |  | **Source tag-name Destination.**<br><br>[**Source** relates to **Destination.**] | |
| Bidirectional tagged |  | **A a-to-b tag B.**<br>**B b-to-a tag A.** | |
| Reciprocal tagged<br><br>[Reciprocal null tagged] |  | **A** and **B** are **reciprocal tag.**<br><br>[**A** and **B** are related.] | |

# OPM Behavior

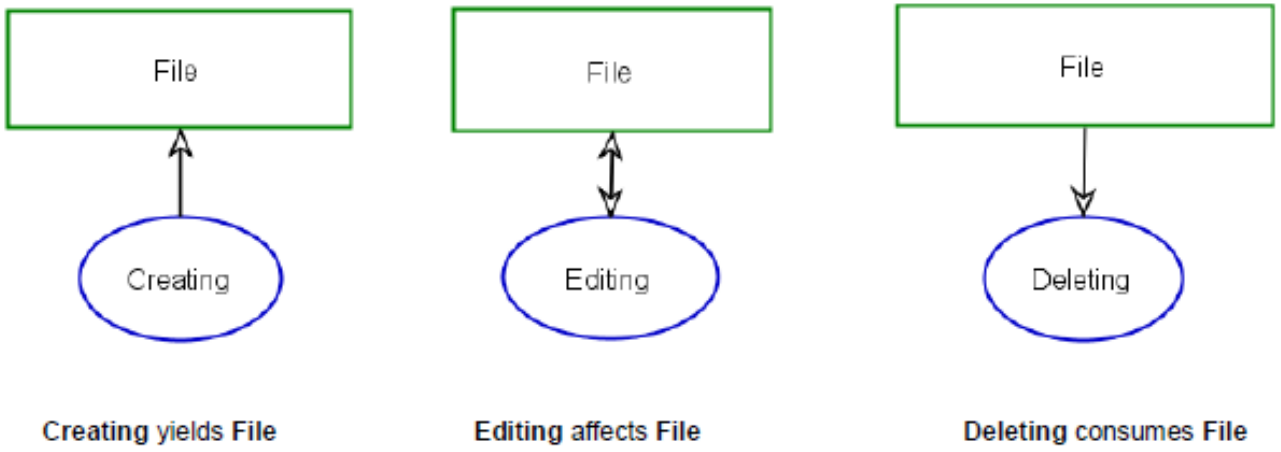
Procedural Links

- **Procedural Link** is a link that specifies a **dynamic aspect** of the system by connecting na object (or one of its states) and a process.

- Procedural links symbolize the behavior of the modeled system.
- Three types:
  - Enabling links: link a **process** to an **object** that enables the **process** but is not affected by it.
  - Transforming links: links a **process** to an **object** that is affected by the **process**.
  - Invocation links: shortcut notation between two consecutive **processes**.

# Transforming Links

- A transformation link is a procedural link that connects a process with an object transformed by the process. Three types:
  - **Consumption**: the linked object is consumed and eliminated by the process.
  - **Result**: the linked object is constructed by the process.
  - **Effect**: the linked object is changed by the process.




**Creating** yields **File** | **Editing** affects **File** | **Deleting** consumes **File**

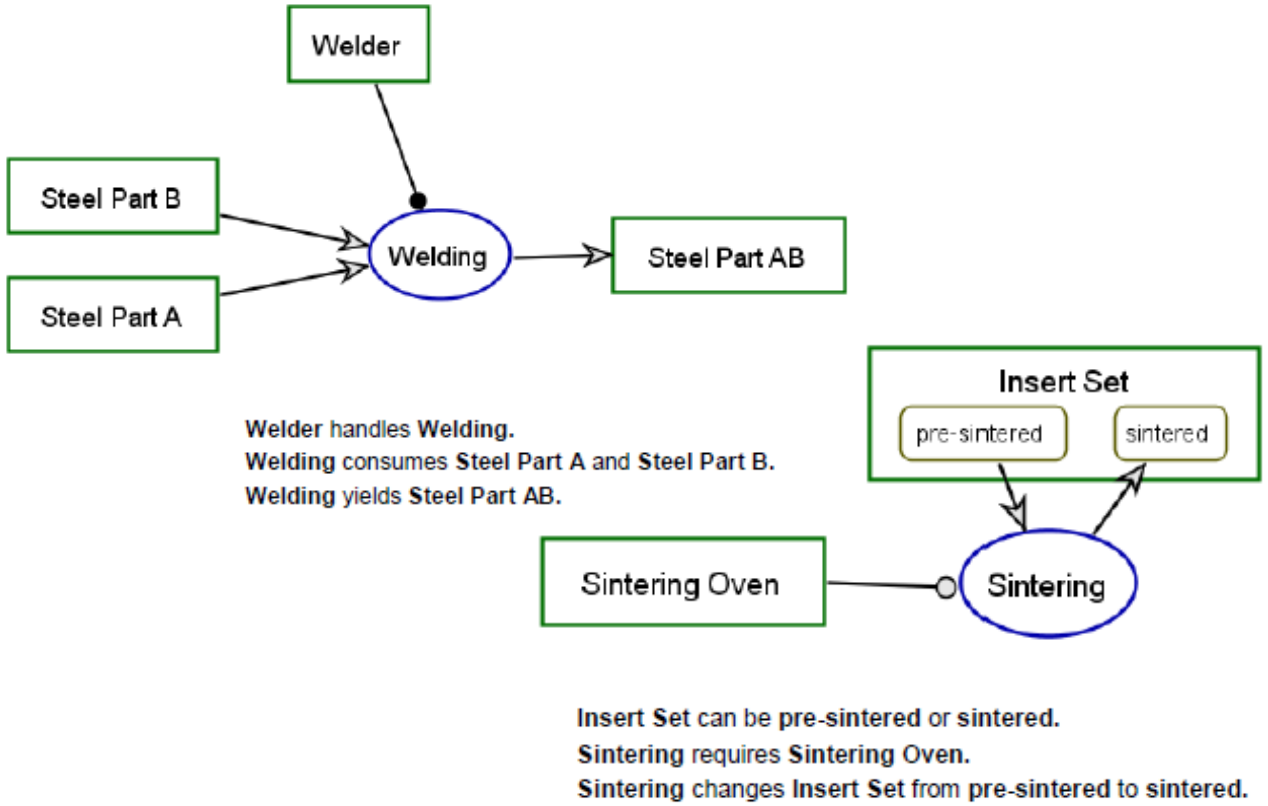
The syntax of a consumption link OPL sentence shall be: ***Processing consumes Consumee*.**

The syntax of a result link OPL sentence shall be: ***Processing yields Resultee*.**

The syntax of an effect link OPL sentence shall be: ***Processing affects Affectee*.**

# Enabling Links

- An enabling link is a procedural link that connects a process with an enabler object of that process. Two types:
  - **Agent**: an enabler who is a human or a group of humans.
  - **Instrument**: a non-human enabler

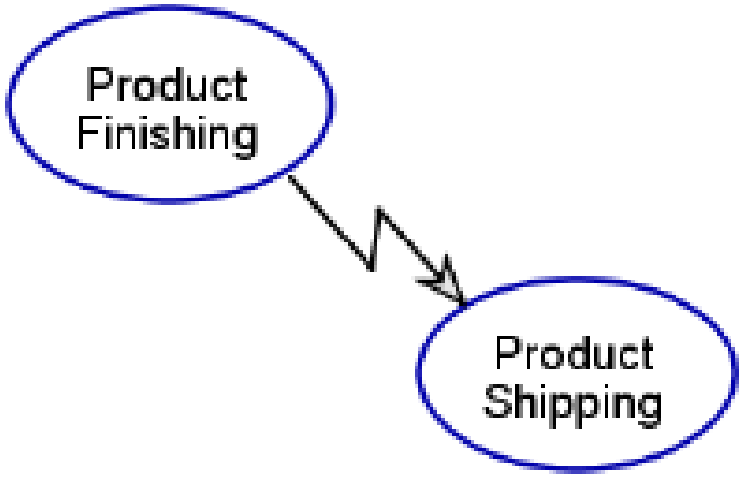



The syntax of an agent link OPL sentence shall be: ***Agent handles Processing*.**

The syntax of an instrument link OPL sentence shall be: ***Processing requires Instrument*.**

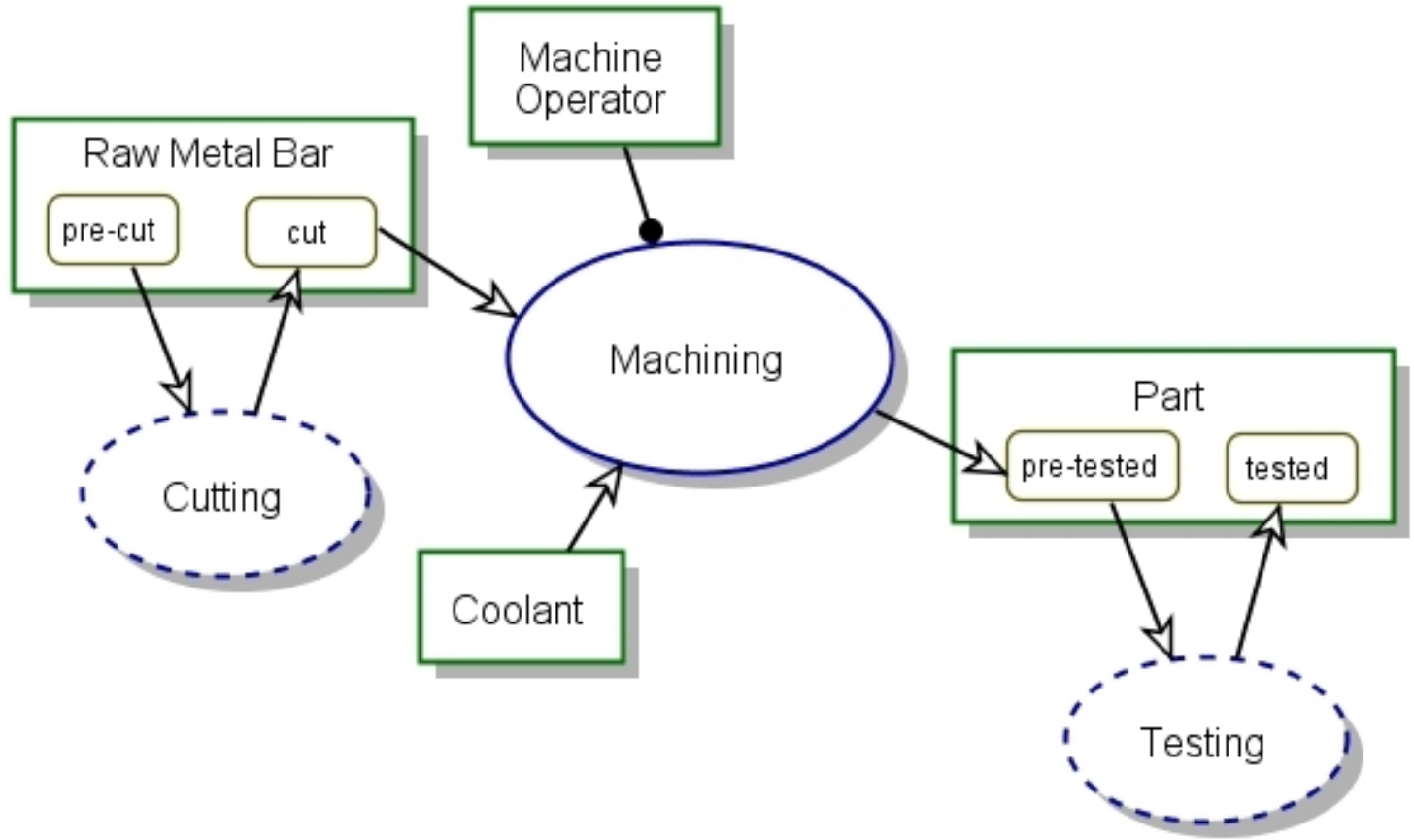
# Invocation Link

- By definition, a process must transform an object. But sometimes the result of a process is not significant to the system and may be ignored. However, the result of the process is significant to a consecutive process.
- The invocation link provides a shortcut to bypass the modeling of the irrelevant object.
  - The syntax of an invocation link OPL sentence shall be: ***Invoking-process invokes invoked-process***.
  - The syntax of a self-invocation link OPL sentence shall be: ***Invoking-process invokes itself***.



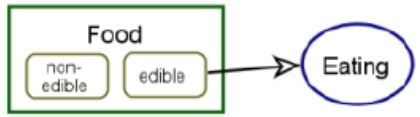
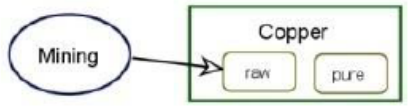
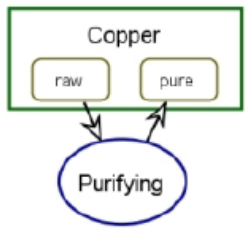
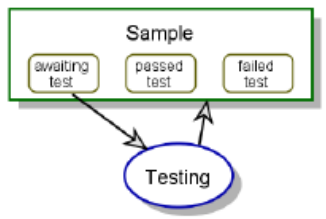
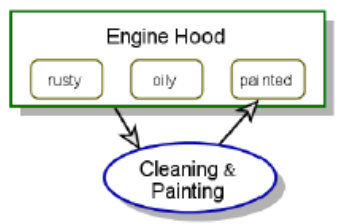

**Product Finishing** invokes **Product Shipping**.

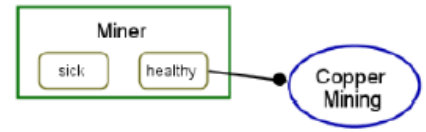
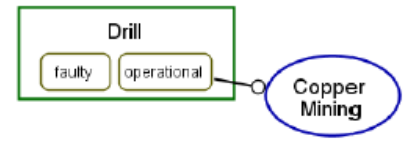
# States can be also used in some relations




**Raw Metal Bar** is **physical.**
**Raw Metal Bar** can be **pre-cut** or **cut.**
**Machine Operator** is **physical.**
**Coolant** is **physical.**
**Machining** is **physical.**
**Machining** requires **Coolant.**
**Machine Operator** handles **Machining.**
**Part** is **physical.**
**Part** can be **pre-tested** or **tested.**
**Testing** is **environmental** and **physical.**
**Cutting** changes **Raw Metal Bar** from **pre-cut** to **cut.**
**Machining** consumes **Raw Metal Bar.**
**Machining** yields **pre-tested Part.**
**Testing** changes **Part** from **pre-tested** to **tested.**

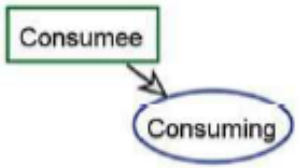
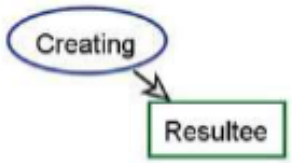
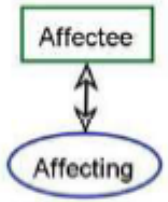
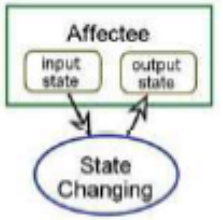
Remember to review text from this model ( Machining consumes Coolant. )

| Name | Semantics | Sample OPD & OPL | Source | Destination |
|---|---|---|---|---|
| **State-specified consumption link** | The process consumes the object if and only if the object is in the specified state. |  **Eating** consumes **edible Food.** | consumee state | process |
| **State-specified result link** | The process generates the object in the specified state. |  **Mining** yields **raw Copper.** | process | resultee state |
| **Input-output-specified effect link pair** (consisting of one state-specified *input link* and one state-specified *output link*) | The process changes the object from a specified input state via the *input link* to a specified output state via the *output link*. |  **Purifying** changes **Copper** from **raw** to **pure.** | affectee source state | affecting process |
| | | | affecting process | affectee destination state |
| **Input-specified effect link pair** (consisting of one state-specified *input link* and one state-unspecified *output link*) | The process changes the object from a specified input state to any output state. |  **Testing** changes **Sample** from **awaiting test.** | affectee source state | affecting process |
| | | | affecting process | affectee |
| **Output-specified effect link pair** (consisting of one state-unspecified *input link* and one state-specified *output link*) | The process changes the object from any input state to a specified output state. |  **Cleaning & Painting** changes **Engine Hood** to **painted.** | affectee | affecting process |
| | | | affecting process | affectee destination state |

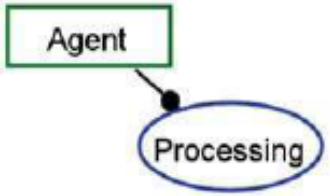
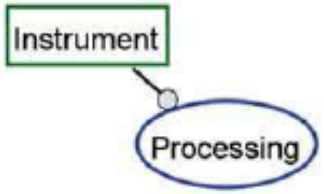
| Name | Semantics | Sample OPD & OPL | Source | Destination |
|---|---|---|---|---|
| **State-specified agent link** | The human agent enables the process provided she is at the specified state. |  **Healthy Miner** handles **Copper Mining.** | agent state | enabled process |
| **State-specified instrument link** | The process requires the instrument at the specified state. |  **Copper Mining** requires **operational Drill.** | instrument state | enabled process |

# Summary

Procedural transforming links

| consumption link | result link | effect link | in-out link pair |
|---|---|---|---|
|  |  |  |  |
| **Consuming** consumes **Consumee.** | **Creating** yields **Resultee.** | **Affecting** affects **Affectee.** | **State Changing** changes **Affectee** from **input state** to **output state.** |

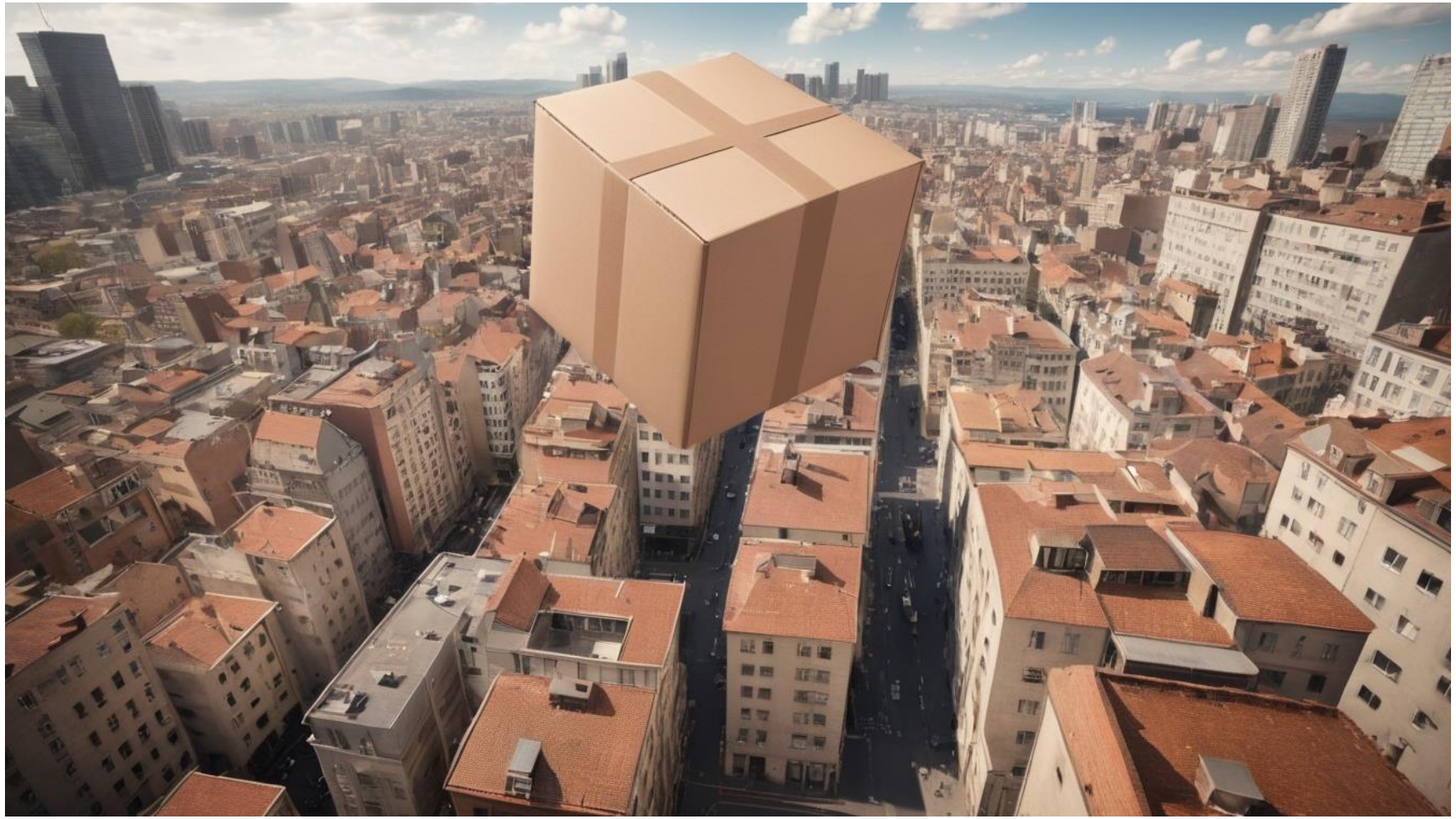
Procedural enabling links

| agent link | instrument link |
|---|---|
|  |  |
| **Agent** handles **Processing.** | **Processing** requires **Instrument.** |

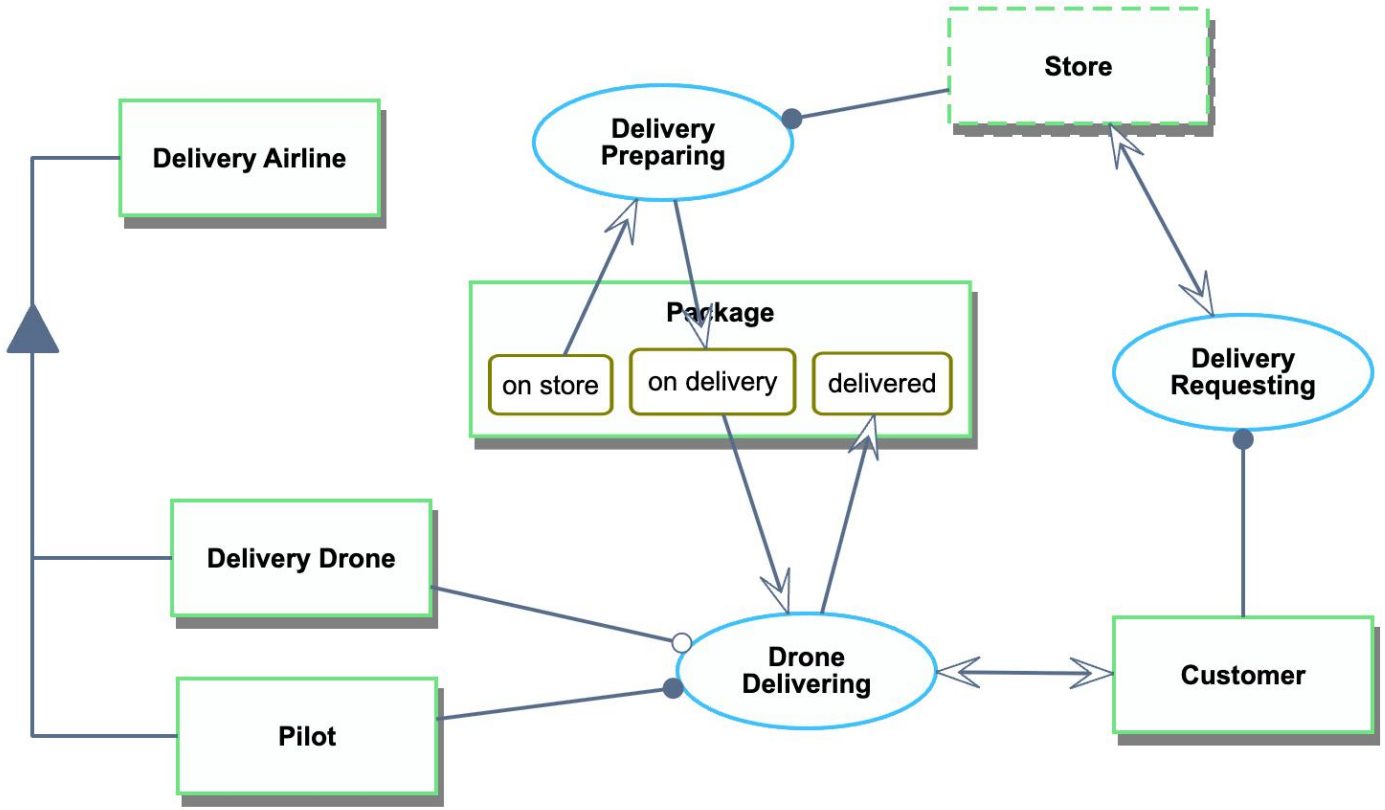
# One-page Simple Example

# Walking through one example

- **Aerial package delivery**



# Walking through one single page example



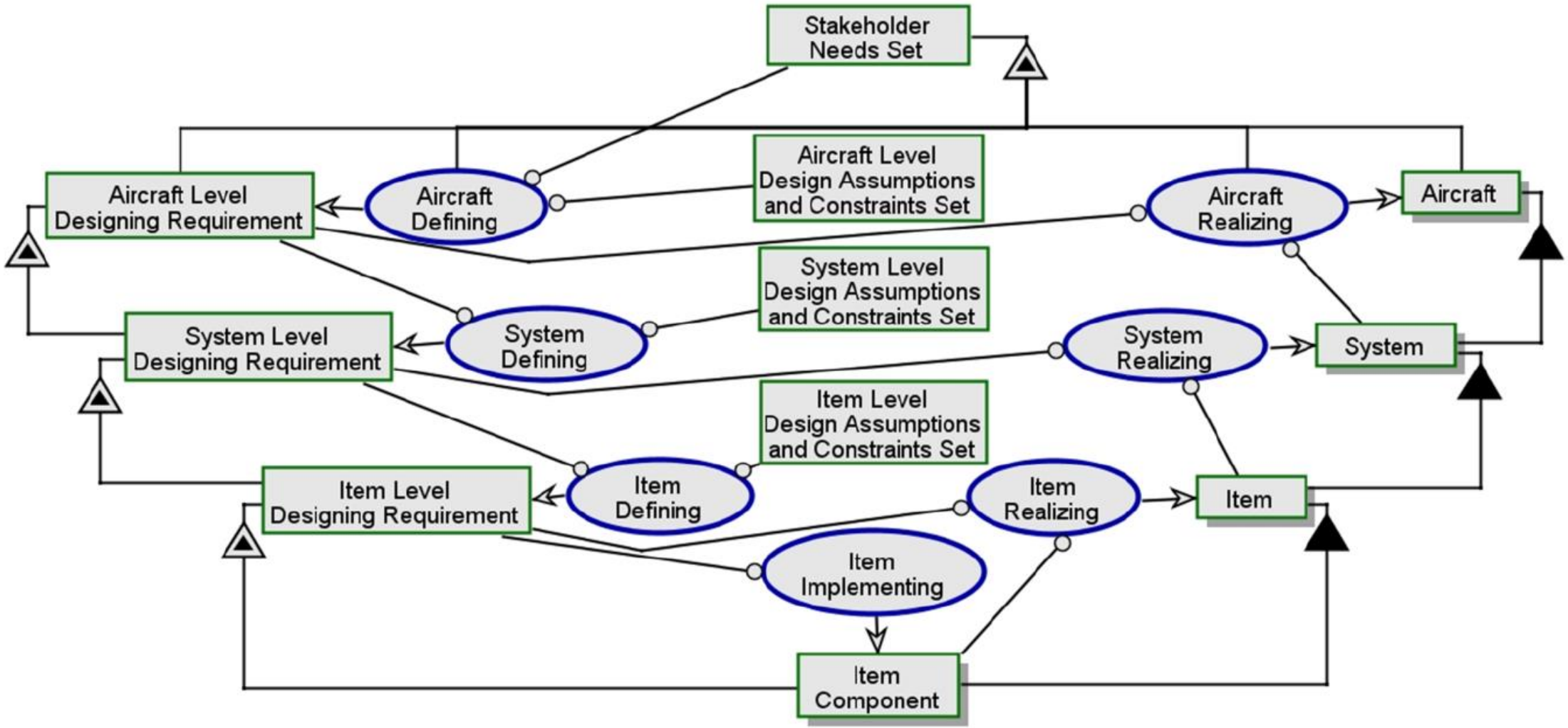

1. **Delivery Airline** is a physical object.
2. **Delivery Drone** is a physical object.
3. **Pilot** is a physical object.
4. **Package** is a physical object.
5. **Package** can be **delivered,on delivery** or **on store.**
6. **Customer** is a physical object.
7. **Store** is a physical and environmental object.
8. **Delivery Airline** consists of **Delivery Drone** and **Pilot.**
9. **Drone Delivering** changes **Package** from **on delivery** to **delivered.**
10. **Pilot** handles **Drone Delivering.**
11. **Drone Delivering** requires **Delivery Drone.**
12. **Drone Delivering** affects **Customer.**
13. **Delivery Preparing** changes **Package** from **on store** to **on delivery.**
14. **Store** handles **Delivery Preparing.**
15. **Customer** handles **Delivery Requesting.**
16. **Delivery Requesting** affects **Store.**

# Final Remarks

# Model-Based Systems Engineering for Aircraft Design with Dynamic Landing Constraints Using Object-Process Methodology

Linwen Li[1 * † ‡], Natali Levi Soskin[2 †], Ahmad Jbara[3 †], Moti Karpel[4 ‡] and Dov Dori[5 †]

[*]*Shanghai Aircraft Design and Research Institute, Shanghai, 201210, China*




SOME EXAMPLES

# Some thoughts

- OPM is **simple and powerful to talk with stakeholders** and create the first architectures
- OPM uses **one diagram type** to handle structure and behavior
  - the language vocabulary has only a **couple of symbols** and **semantics to mimic common sketching**.
- OPM allows simple-formal modelling and enables to **control the complexity**.
- OPM is the only MBSE that **simulates CONCEPTS**.
- As it is an ISO, it is **worth a try**.
- OPM **lacks transformational tools** to other domains and a open metamodel (EMF).

# OPM is growing

- OPM main tool is the OPCloud (“web” based)
- It has been highly improved from the OPCat used through the course.
  - Usability is better
  - Allows dynamic behavior
  - IoT connectable through MQTT
  - Socket connection
  - Stereotypes
  - Styling
  - Timing